Alignment Analysis of Sequential Segmentation of Lexicons to Improve Automatic Cognate Detection

Pranav A
Big Data Institute
Hong Kong University of Science and Technology
Hong Kong
cs.pranav.a@gmail.com

Abstract

Ranking functions in information retrieval are often used in search engines to recommend the relevant answers to the query. This paper makes use of this notion of information retrieval and applies onto the problem domain of cognate detection. The main contributions of this paper are: (1) positional segmentation, which incorporates the sequential notion; (2) graphical error modelling, which deduces the transformations. The current research work focuses on classification problem; which is distinguishing whether a pair of words are cognates. This paper focuses on a harder problem, whether we could predict a possible cognate from the given input. Our study shows that when language modelling smoothing methods are applied as the retrieval functions and used in conjunction with positional segmentation and error modelling gives better results than competing baselines, in both classification and prediction of cognates.

1 Introduction

Cognates are a collection of words in different languages deriving from the same origin. The study of cognates plays a crucial role in applying comparative approaches for historical linguistics, in particular, solving language relatedness and tracking the interaction and evolvement of multiple languages over time. A cognate instance in Indo-European languages is given as the word group: *night* (English), *nuit* (French), *noche* (Spanish) and *nacht* (German).

The existing studies on cognate detection involve experiments which distinguish between a pair of words whether they are cognates or non-cognates (Ciobanu and Dinu, 2014; List, 2012a). These studies do not approach the problem of predicting the possible cognate of the target language, if the cognate of the source language is given. For example, given the word *nuit*, could the algorithm predict the appropriate German cognate within the huge German wordlist? This paper tackles this problem by incorporating heuristics of the probabilistic ranking functions from information retrieval. Information retrieval addresses the problem of scoring a document with a given query, which is used in every search engine. One can view the above problem as the construction of a suitable search engine, through which we want to find the cognate counterpart of a word (query) in a lexicon of another language (documents).

This paper deals with the intersection between the areas of information retrieval and approximate string similarity (like the cognate detection problem), which is largely under-explored in the literature. Retrieval methods also provide a variety of alternative heuristics which can be chosen for the desired application areas (Fang et al., 2011). Taking such advantage of the flexibility of these models, the combination of approximate string similarity operations with an information retrieval system could be beneficial in many cases. We demonstrate how the notion of information retrieval can be incorporated into the approximate string similarity problem by breaking a word into smaller units. Regarding this, Nguyen et al. (2016) has argued that segmented words are a more practical way to query large databases of sequences, in comparison with conventional query methods. This further encourages the heuristic attempt at imposing an information retrieval model on the cognate detection problem in this paper.

Our main contribution is to design an information retrieval based scoring function (see section 4) which can capture the complex morphological

shifts between the cognates. We tackled this by proposing a shingling (chunking) scheme which incorporates positional information (see section 2) and a graph-based error modelling scheme to understand the transformations (see section 3). Our test harness focuses not only on distinguishing between a pair of cognates, but also the ability to predict the cognate for a target language (see section 5).

2 Positional Character-based Shingling

This section examines on converting a string into a shingle-set which includes the encodings of the positional information. In this paper, we notify, S as the shingle-set of cognate from the source language and T as the shingle-set of cognate for the target language. The similarity between these split-sets is denoted by $S \cap T$. An example of cognate from the source language, S (Romanian) could be shingle set of the word *rosmarin* and T (Italian) could be *romarin*.

K-gram shingling: Usually, set based string similarity measures are based on comparing overlap between the shingles of two strings. Shingling is a way of viewing a string as a document by considering k characters at a time. For example, the shingle of the word *rosmarin* is created with $k = 2$ as: $S = \{\langle s \rangle r, ro, os, sm, ma, ar, ri, in, n \langle /s \rangle\}$. Here, $\langle s \rangle$ is the start sentinel token and $\langle /s \rangle$ is the stop sentinel token. For the sake of simplicity, we have ignored sentinel tokens; which transforms into: $S = \{r, ro, os, sm, ma, ar, ri, in, n\}$. This method splits the strings into smaller k-grams without any positional information.

2.1 Positional Shingling from 1 End

We argue that the unordered *k-grams* splitting could lead to an inefficient matching of strings since a shingle set is visualized as the bag-of-words method. Given this, we propose a positional k-gram shingling technique, which introduces position number in the splits to incorporate the notion of the sequence of the tokens. For example, the word *rosmarin* could be position-wise split with $k = 2$ as: $S = \{1r, 2ro, 3os, 4sm, 5ma, 6ar, 7ri, 8in, 9n\}$.

Thus, the member *4sm* means that it is the fourth member of the set. The motivation behind this modification is that it retains the positional information which is useful in probabilistic retrieval ranking functions.

2.2 Positional Shingling from 2 Ends

The main disadvantage of the positional shingling from single end is that any mismatch can completely disturb the order of the rest, leading to low similarity. For example, if the query is *rosmarin* with cognate *romarin*, the corresponding split sets would be $\{1r, 2ro, 3os, 4sm, 5ma, 6ar, 7ri, 8in, 9n\}$ and $\{1r, 2ro, 3om, 4ma, 5ar, 6ri, 7in, 8n\}$. The order of the members after *2ro* is misplaced, thus this will lead to low similarity between two cognates. Only $\{1r, 2ro\}$ is common between the cognates. Considering this, we propose positional shingling from both ends, which is robust against such displacements.

We attach a position number to the left if the numbering begins from the start, and to the right if the numbering begins from the end. Then the smallest position number is selected between the two position numbers. If the position numbers are equal, then we select the left position number as a convention. Figure 1 gives an exemplification of this algorithm illustrated with splits of *romarin* and *rosmarin*.

Figure 1: The process of positional tokenisation from both ends. On the left, algorithm segments the Romanian word *romarin* into the split-set $\{1r, 2ro, 3om, 4ma, ar4, ri3, in2, n1\}$. On the right, the algorithm segments *rosmarin* into $\{1r, 2ro, 3os, 4sm, 5ma, ar4, ri3, in2, n1\}$.

In Figure 1, split sets of *rosmarin* and *romarin* are shown. After taking intersection of them, we get $\{1r, 2ro, ar4, ri3, in2, n1\}$, indicating a higher similarity.

3 Graphical Error Modelling

Once shingle sets are created, common overlap set measures like set intersection, Jaccard (Järvelin et al., 2007), XDice (Brew et al., 1996) or TF-IDF (Wu et al., 2008) could be used to measure similarities between two sets. However, these methods only focus on similarity of the two strings. For cognate detection, it is crucial to understand how substrings are transformed from source language to target language. This section discusses on how to view this "dissimilarity" by creating a graphical error model.

Algorithm 1 explicates the process of graphical error modelling. For illustration purposes, we visualize the procedure via a Romanian-Italian cognate pair (*mesia*, *messia*). If the source language is Romanian, then $S = \{1m, 2me, 3es, si3, ia2, a1\}$, which is the split-set of *mesia*. Let the target language by Italian. Then the split-set of the Italian word *messia*, denoted as T, will be $\{1m, 2me, 3es, 4ss, si3, ia2, a1\}$. Thus $|S \cap T|$ is the number of common terms. Thus the term matches are, $S \cap T = \{1m, 2me, 3es, si3, ia2, a1\}$. We are interested in examining the "dissimilarity", which are the leftover terms in the sets. That means, we need to infer a certain pattern from leftover sets, which are $S - \{S \cap T\}$ and $T - \{S \cap T\}$. Thus we can draw mappings to gather information of the corrections. Let *top* and *bottom* be the **ordered sets** referring to $S - \{S \cap T\}$ and $T - \{S \cap T\}$ respectively. Referring to the example, $T - S \cap T = \{4ss\}$, a *bottom* set. Similarly, $S - S \cap T = \{\}$, a *top* set. Then we follow instructions given in algorithm 1.

Algorithm 1: Graphical Error Model

Graphical Error Model takes two split sets generated by the shingling variants, namely *top* and *bottom*. The objective is to output a graphical structure showing connections between members of the *top* and the *bottom* sets.

1. **Initialization of the *top* and *bottom*:** If the given sets *top* and *bottom* are empty, we initialize them by inserting an empty token (ϕ) into those sets.
   Running example: This step transforms *top* set as $\{\phi\}$ and *bottom* as $\{4ss\}$.
2. **Equalization of the set cardinalities:** The cardinalities of the sets *top* and *bottom* made equal by inserting empty tokens (ϕ) into the middle of the sets.

Running example: The set cardinalities of *top* and *bottom* were already equal. Thus the output of this step are *top* set as $\{\phi\}$ and *bottom* as $\{4ss\}$.

3. **Inserting the mappings of the set members into the graph:** The empty graph is initialized as $graph = \{\}$. The directed edges are generated, originating from every set member of *top* to every set member of *bottom*. This results in a complete directed bipartite graph between *top* and *bottom* sets. Each edge is assigned a probability $P(e)$ which is discussed in a later section.
   Running example: The output of this step would be complete directed bipartite graph between *top* and *bottom* sets which is $\{\phi \rightarrow 4ss\}$
   One more example is provided in figure 2.

Intuition: The edges created as the result of this graph could be used for probabilistic calculations which are detailed more in section 4.2. Intuitively, $\phi \rightarrow 4ss$ means that if the letter *s* is added at position 4 of the word of the source *mesia*, then one could get the target word *messia*.
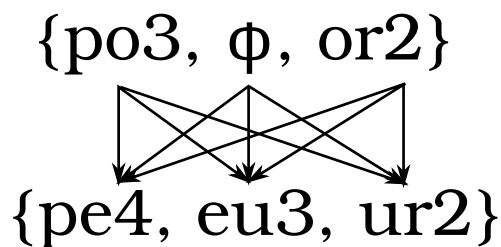

Figure 2: The figure shows the bipartite graph output of the algorithm when the source cognate is *stupor* and the target cognate is *stupeur*.

4 Evaluation Function

The design of our evaluation function focuses on two main properties: set based similarity (see section 4.1) and probabilistic calculation through graphical model (see section 4.2)

4.1 Similarity Function

Usually, the computation of similarity between two sets is done by metrics like Jaccard, Dice and XDice (Brew et al., 1996). Dynamic programming based methods like edit distance and LCSR (Longest Common Subsequence Ratio,

([Melamed, 1999](#)) are also often used to calculate similarity between two strings. Ranking functions incorporate more complex but necessary features which are needed to distinguish between the documents.

In this paper, we use BM25 and a Dirichlet smoothing based ranking function to compute the similarity. BM25 considers term-frequency, inverse document frequency and length normalization based penalization features for similarity calculations. Dirichlet smoothing function ([Robertson and Zaragoza, 2009](#)) makes use of language modelling features and tunable parameter which aids in Bayesian smoothing of unseen shingles in the split sets ([Blei et al., 2003](#)).

4.2 Error Modelling Function

The information of the common morphological transformations for cognates between two different languages helps in determining if a pair of words could be cognates. Based on the graphs of cognate pairs between Italian and Romanian (section 3), which models the morphological shifts between the cognates in the two languages, we define an error modelling function on any pair of words from the two languages. The split set from the source language is denoted by S and target language by T, then probabilistic function would be:

$$\pi(S, T) = \frac{1}{|G(S,T)|} \sum_{e \in G(S,T)} P(e)^q$$

where $G(S,T)$ is the constructed graph of S and T, the strength parameter is called q here with the range of $(0, \infty)$, and $P(e)$ is the probability of edge e to occur in between two cognates, which is estimated by its frequency of being observed in the graphs of cognate pairs in the training set.

Figure 3 illustrates the aggregation of edges in the graph and figure 4 shows the final output of the error modelling function after normalizing.

$\pi(S, T)$ is called the error modelling function defined for the word pair, which is an intuitive calculation of probabilty between a pair of cognates through estimating their transformations. q is a tunable parameter that controls the effect of the probabilistic frequencies $P(e)$ observed in the training set, often useful in avoiding overfitting. $\frac{1}{|G(S,T)|}$ is the normalization factor to allow us to compare the quantity across different word pairs.
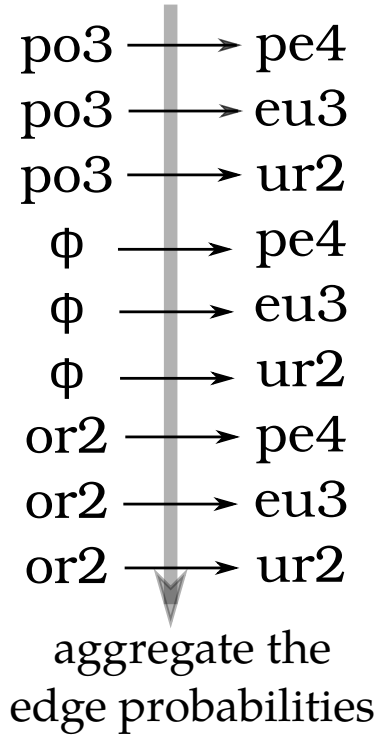

Figure 3: From the graph created in figure 2, we calculate the probabilities of each edge (by computing frequencies and smoothing) and then aggregate all the probabilities of edges in the graph.
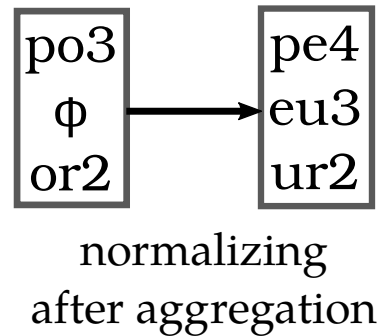

Figure 4: After aggregating, we normalize the sum and the graph conversion score is outputted.

4.3 Combining Error Modelling and Similarity Function metrics

In this subsection, we merge the notion of similarity and dissimilarity together. We combine a set-based similarity function (discussed in section 4.1) and the error modelling function (discussed in section 4.2) into a score function by a weighted sum of them, which is,

$$score(S, T) = \lambda \times sim(S, T) + (1 - \lambda) \times \pi(S, T)$$

where $\lambda \in [0, 1]$ is a weight based hyperparameter, $sim(S, T)$ is a set-based similarity between S and

T, and $\pi(S, T)$ is the graphical error modelling function defined above.

5 Test Harness

Table 1 summarizes the results of the experimental setup. The elements of test harness are mentioned as following:

5.1 Setup Description

Dataset: The experiments in this paper are performed on the dataset used by Ciobanu *et al* (2014). The dataset consists 400 pairs of cognates and non-cognates for Romanian-French (Ro - Fr), Romanian-Italian (Ro - It), Romanian-Spanish (Ro - Es) and Romanian-Portuguese (Ro - Pt). The dataset is divided into a 3:1 ratio for training and testing purposes. Using cross-validation, hyperparameters and thresholds for all the algorithms and baselines were tuned accordingly in a fair manner. **Experiments:** Two experiments are included in test harness.

1. We provide a pair of words and the algorithms would aim to detect whether they are cognates. Accuracy on the test set is used as a metric for evaluation.

2. We provide a source cognate as the input and the algorithm would return a ranked list as the output. The efficiency of the algorithm would depend on the rank of the desired target cognate. This is measured by MRR (Mean Reciprocal Rank), which is defined as, $MRR = \sum_{i=1}^{|dataset|} \frac{1}{rank_i}$, where $rank_i$ is the rank of the true cognate in the ranked list returned to the ith query.

5.2 Baselines

String Similarity Baselines: It is intuitive to compare our methods with the prevalent string similarity baselines since the notions behind cognate detection and string similarity are almost similar. Edit Distance is often used as the baseline in the cognate detection papers (Melamed, 1999). This computes the number of operations required to transform from source to target cognate. We have also incorporated XDice (Brew et al., 1996), which is a set based similarity measure that operates between shingle set between two strings. Hidden alignment conditional random fields (CRF) are often used in transliteration which serves as the generative sequential model to compute the probabilities between the cognates, which is analogous

to learnable edit distance (McCallum et al., 2012). Among these baselines, CRF performs the best in accuracy and MRR.

Orthographic Cognate Detection: Papers related to this notion usually take alignment of substrings which in classifier like support vector machines (Ciobanu and Dinu, 2015, 2014) or hidden markov models (Bhargava and Kondrak, 2009). We included Alina *et al* as the baseline (2014), which employs the dynamic programming based methods for sequence alignment following which features were extracted from the mismatches in the word alignments. These features are plugged into the classifier like SVM which results in decent performance on accuracy with an average of 84%, but only 16% on MRR. This result is due to the fact that a large number of features leads to overfitting and scoring function is not able to distinguish the appropriate cognate.

Phonetic Cognate Detection: Research in automatic cognate identification using phonetic aspects involve computation of similarity by decomposing phonetically transcribed words (Kondrak, 2000), acoustic models (Mielke, 2012), phonetic encodings (Rama, 2015), aligned segments of transcribed phonemes (List, 2012b). We implemented Rama's research (2015), which employs a Siamese convolutional neural network to learn the phonetic features jointly with language relatedness for cognate identification, which was achieved through phoneme encodings. Although it performs well on accuracy, it shows poor results with MRR, possibly the reason as same as SVM performance.

5.3 Ablation experiments

We experiment with the variables like length of substrings, ranking functions, shingling techniques, and graphical error model, which are detailed in the Table 1. Amongst the shingling techniques, we found that character bigrams with 2-ended positioning give better results. Adding trigrams to the database does not give major effect on the results. The results clearly indicate that adding graphical error model features greatly improve the test results. Amongst the ranking functions, Dirichlet smoothing tends to give better results, possibly due to the fact that it requires fewer parameters to tune and is able to capture the sequential data (like substrings) better than other ranking functions (Fang et al., 2011). The hy-

Algorithms	Ro - It		Ro - Fr		Ro - Es		Ro - Pt	
	Acc	MRR	Acc	MRR	Acc	MRR	Acc	MRR
Edit Distance (Melamed, 1999)	0.53	0.11	0.52	0.13	0.58	0.15	0.54	0.13
XDice (Brew et al., 1996)	0.54	0.19	0.53	0.14	0.59	0.16	0.53	0.14
SVM with Orthographic Aligment (Ciobanu and Dinu, 2014)	0.81	0.18	0.87	0.15	0.86	0.16	0.73	0.14
Phonetic Encodings in CNN (Rama, 2016)	0.69	0.21	0.78	0.17	0.77	0.19	0.66	0.15
Hidden alignment CRF (McCallum et al., 2012)	0.84	0.51	0.85	0.48	0.84	0.50	0.71	0.45

Shingling Technique	Ranking Function	Ro - It		Ro - Fr		Ro - Es		Ro - Pt	
Bigram, 0-ended	TF-IDF	0.54	0.18	0.52	0.14	0.59	0.15	0.55	0.11
Bigram, 1-ended	TF-IDF	0.59	0.19	0.54	0.18	0.60	0.18	0.57	0.12
Bigram, 2-ended	TF-IDF	0.64	0.25	0.63	0.21	0.68	0.22	0.65	0.17
(Bi + Tri)-gram, 2-ended	TF-IDF	0.64	0.25	0.64	0.21	0.57	0.22	0.65	0.18
Bigram, 2-ended	BM25	0.84	0.40	0.87	0.37	0.86	0.34	0.73	0.35
Bigram, 2-ended	Dirichlet	0.84	0.41	0.86	0.38	0.86	0.39	0.74	0.36
Bigram, 2-ended	BM25 + Graphical Error Model	0.87	0.64	0.89	0.51	0.86	0.54	0.78	0.55
Bigram, 2-ended	Dirichlet + Graphical Error Model	**0.88**	**0.67**	**0.89**	**0.59**	**0.87**	**0.60**	**0.80**	**0.58**

Table 1: Results on the test dataset. The upper half denotes the baselines used and the lower half describes our ablation experiments. For the experiment 1, we evaluate using the accuracy (Acc) metric. We used MRR (Mean Reciprocal Rank) for describing the second experiment. Higher scores signify the better performance. The maximum value possible is 1.0. It is worth noting that for the classification problem (experiment 1), our algorithm has slight improvement. However for the recommendation problem (experiment 2), our algorithm shows massive improvement. The thresholds, hyper-parameters, source code and sample Python notebooks are available at our github repository: https://github.com/pranav-ust/cognates

perparameter λ mentioned in the section 4.3, was tuned around 0.6, which shows the 60% contribution by the similarity function and 40% contribution by the dissimilarity. Overall, the combination of bigrams with 2-ended positional shingling, graphical error modelling with Dirichlet ranking function gives the best performance with an average of 86% on accuracy metric and 60% on MRR.

6 Conclusions

We approach towards the harder problem where the algorithm aims to find a target cognate when a source cognate is given. Positional shingling outperformed non-positional shingling based methods, which demonstrates that inclusion of positional information of substrings is rather important. Addition of graphical error model boosted the test results which shows that it is crucial to add dissimilarity information in order to capture the transformations of the substrings. Methods whose scoring functions rely only on complex machine learning algorithms like CNN or SVM tend to give worse results when searching for a cognate.

Acknowledgements

References

Aditya Bhargava and Grzegorz Kondrak. 2009. Multiple word alignment with profile hidden markov models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, pages 43–48. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Chris Brew, David Mckelvie, and Buccleuch Place. 1996. Word-pair extraction for lexicography.

Alina Maria Ciobanu and Liviu P Dinu. 2014. Automatic detection of cognates using orthographic alignment. In *ACL (2)*, pages 99–105.

Alina Maria Ciobanu and Liviu P Dinu. 2015. Automatic discrimination between cognates and borrowings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 431–437.

Hui Fang, Tao Tao, and Chengxiang Zhai. 2011. Diagnostic evaluation of information retrieval models. *ACM Trans. Inf. Syst.*, 29(2):7:1–7:42.

Anni Järvelin, Antti Järvelin, and Kalervo Järvelin. 2007. s-grams: Defining generalized n-grams for information retrieval. *Information Processing & Management*, 43(4):1005–1019.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 288–295. Association for Computational Linguistics.

Johann-Mattis List. 2012a. Lexstat: Automatic detection of cognates in multilingual wordlists. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, EACL 2012, pages 117–125, Stroudsburg, PA, USA. Association for Computational Linguistics.

Johann-Mattis List. 2012b. Lexstat: Automatic detection of cognates in multilingual wordlists. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pages 117–125. Association for Computational Linguistics.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2012. A conditional random field for discriminatively-trained finite-state string edit distance. *arXiv preprint arXiv:1207.1406*.

I. Dan Melamed. 1999. Bitext maps and alignment via pattern recognition. *Comput. Linguist.*, 25(1):107–130.

Jeff Mielke. 2012. A phonetically based metric of sound similarity. *Lingua*, 122(2):145–163.

Ken Nguyen, Xuan Guo, and Y Pan. 2016. *Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications*.

Taraka Rama. 2015. Automatic cognate identification with gap-weighted string subsequences. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*, pages 1227–1231.

Taraka Rama. 2016. Siamese convolutional networks based on phonetic features for cognate identification. *CoRR*, abs/1605.05172.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting tf-idf term weights as making relevance decisions. *ACM Trans. Inf. Syst.*, 26(3):13:1–13:37.