Detecting English-French Cognates Using Orthographic Edit Distance

Qiongkai Xu[1,2], **Albert Chen**[1], **Chang Li**[1]
[1]The Australian National University, College of Engineering and Computer Science
[2]National ICT Australia, Canberra Research Lab
{xuqiongkai, u5708995, spacegoing}@gmail.com

Abstract

Identification of cognates is an important component of computer assisted second language learning systems. We present a simple rule-based system to recognize cognates in English text from the perspective of the French language. At the core of our system is a novel similarity measure, orthographic edit distance, which incorporates orthographic information into string edit distance to compute the similarity between pairs of words from different languages. As a result, our system achieved the best results in the ALTA 2015 shared task.

1 Introduction

Cognates words are word pairs that are similar in meaning, spelling and pronunciation between two languages. For example, "*age*" in English and "*âge*" in French are orthographically similar while "father" in English and "Vater" in German are phonetically similar. There are three types of cognates: true cognates, false cognates and semi-cognates. True cognates may have similar spelling or pronunciation but they are mutual translations in any context. False cognates are orthographically similar but have totally different meanings. Semi-cognates are words that have the same meaning in some circumstances but a different meaning in other circumstances. Finding cognates can help second language learners leverage their background knowledge in their first language, thus improving their comprehension and expanding their vocabulary.

In this paper, we propose an automatic method to identify cognates in English and French with the help of the Google Translator API[1]. Our method calculates the similarity of two words based solely on the sequences of characters involved. After exploring n-gram similarity and edit distance similarity, we propose an orthographic edit distance similarity measure which leverages orthographic information from source language to target language. Our approach achieved first place in the ALTA 2015 shared task.

2 Related Work

There are many ways to measure the similarity of words from different languages. Most popular ones are surface string based similarity, i.e. n-gram similarity and edit distance. An n-gram is a contiguous sequence of n items, normally letters, from a given sequence. There are many popular measures that use n-grams such as DICE (Brew et al., 1996), which uses bi-grams, and Longest Common Subsequence Ratio (LCSR) (Melamed, 1999). LCSR was later found to be a special case of n-gram similarity by Kondrack (Kondrak, 2005), who developed a general n-gram framework. He provided formal, recursive definitions of n-gram similarity and distance, together with efficient algorithms for computing them. He also proved that in many cases, using bi-grams is more efficient than using other n-gram methods. Since LCSR is only a tri-gram measure, using bi-gram similarity and distance can easily outperform LCSR in many cases.

Instead of computing common n-grams, word similarity can be also measured using edit distance. The edit distance between two strings is the minimum number of operations that are needed to transform one string into another. When calculating the edit distance, normally three operations are considered: removal of a single character, insertion of a single character and substitution of one character with another one. Levenshtein defined each of these operations as having unit cost except for substitution (Levenshtein, 1966). Other suggestions have been made to add more opera-

[1]https://code.google.com/p/google-api-translate-java/

Language	Sentence
English	We have to do it out of respect.
French	Nous devons le faire par respect

Table 1: Phrase alignment of machine translation.

tions like merge and split operations in order to consider adjacent characters (Schulz and Mihov, 2002). The algorithm was improved by Ukkonen using a dynamic programming table around its diagonal making it linear in time complexity (Ukkonen, 1985).

3 System Framework

To tackle the ALTA 2015 shared task[2], we propose a system consisting of the following steps:

- Step 1: Translate source words (English) into target words (French). Filter out meaningless words or parts of words.
- Step 2: Calculate the similarity score of all word pairs. Search the best threshold and decide if word pairs are cognates.

3.1 Cognate Candidates Generation

Since there is no aligned French corpus provided in this task, we need to generate cognate candidates by using a machine translator. One approach is to translate English sentences into French sentences followed by extracting the aligned words. Although this approach makes use of the words' context, its quality depends on both the quality of the translator and the word alignment technology. Table 1 shows an example of machine translation and phrase alignment results. We find that "do" (faire) and "it" (le) are in a different order when translated into French. We work around this by translating each sentence word by word using the Google Translator API. A benefit of this approach is that we can cache the translation result of each word, making the system more efficient. The total time of calling the translator API is reduced from more than 22,000 to less than 5,600 in the training and testing sets.

Due to the differences between French and English, an English word (a space-separated sequence of characters) may be translated to more than one word in French. For example, Google Translator translates "language's" to "la langue de". To facilitate the process of determining whether "language's" is a cognate in French and English, we first filter out the "'s" from the English word and the "la" and the "de" from the translation. We can then calculate the similarity of "language" and "langue". More generally, we filter out the definite articles "le", "la" and "les" and the preposition "de" from the phrase given by the translator.

[2]http://www.alta.asn.au/events/alta2015/task.html

SL:	$len(S) - n + 1$
Max:	$\max\{len(S) - n + 1, len(T) - n + 1\}$
Sqrt:	$\sqrt{(len(S) - n + 1)(len(T) - n + 1)}$

Table 2: Normalization factor for n-gram similarity.

SL:	$len(S)$
Max:	$\max\{len(S), len(T)\}$
Sqrt:	$\sqrt{len(S)len(T)}$

Table 3: Normalization factor for edit distance similarity.

3.2 N-gram and Edit Distance

For character-level n-gram distance, we calculate the number of common n-gram sequences in source S and target T and then divide by L (the normalization factor) to obtain the normalized n-gram distance similarity:

$$n\_sim(S,T) = \frac{|n\text{-}gram(S) \cap n\text{-}gram(T)|}{L}.$$

We consider three candidates for L: source length (SL), maximum length of S and T (Max), and geometric mean of S and T length (Sqrt) (Table 2).

We calculate the edit distance (Levenshtein distance), from $S = \{s_1, s_2, \ldots, s_n\}$ to $T = \{t_1, t_2, \ldots, t_m\}$ using dynamic programming. The following recursion is used:

$$d_{i,j} = \begin{cases} d_{i-1,j-1} & \text{if } s_i = t_j \\ \min\{d_{i-1,j}, d_{i,j-1}\} & \text{if } s_i \neq t_j \end{cases}$$

where $d_{i,j}$ is the edit distance from $s_{1,i}$ to $t_{1,j}$. Then the similarity score is

$$l\_sim(S,T) = 1 - \frac{d_{n,m}}{L}$$

where L is the normalization factor. Again, we consider three values for L: SL, Max, Sqrt (Table 3).

Instead of using a machine learning algorithm to determine word similarity, we focus on the most promising feature which is edit distance similarity. We further explore this approach and propose a novel similarity measure. A grid search algorithm is utilized to find the best threshold for our system and which works efficiently.

3.3 Edit Distance with Orthographic Heuristic Rules

Although traditional edit distance similarity can figure out cognates in most cases, orthographic information is not utilized properly. We propose an orthographic edit distance similarity which is used to measure the similarity of each pair. We first generate a map that associates common English pieces to French pieces and allows us to ignore diacritics. Suffixes like “k” and “que” are often a feature of cognates in English and French (e.g. “disk” and “disque”). Mapping “e” to “é”, “è” and “ê” helps in finding “system” (English) and “système” (French) as cognates (the accents affect the pronunciation of the word).

If the characters are the same in the two words, the edit distance is zero. Otherwise, we add a penalty, $\alpha \in [0, 1]$, to the edit distance if the suffix of length k of the first i characters of the English word maps to the suffix of length l of the first j characters of the French word. α is set to 0.3 according to our experimentation.

$$d_{i,j} = \min \begin{cases} d_{i-1,j-1} & \text{if } s_i = t_j \\ d_{i-k,j-l} + \alpha & \text{if } \{s_{i-k+1}, \ldots, s_i\} \\ & \to \{t_{j-l+1}, \ldots, t_j\} \\ \{d_{i-1,j}, d_{i,j-1}\} & \text{elsewhere} \end{cases}$$

All orthographic heuristic rules (map) are illustrated in Table 4.

$$e\_sim(S,T) = 1 - \frac{d_{n,m}}{L}$$

The normalization factor is the same as the one used in Section 3.2. The pseudocode for calculating the orthographic edit distance is provided in Algorithm 1.

English	French
e	é è ê ë
a	â à
c	ç
i	î ï
o	ô
u	û ù ü
k	que

Table 4: English-French orthographic Heuristic Rules for orthographic edit distance.

L	Precision(%)	Recall(%)	F-1(%)
SL	73.21	76.59	74.86
Max	72.40	**79.94**	75.98
Sqrt	**75.06**	77.31	**76.17**

Table 5: Result of bi-gram similarity on training dataset using different normalization methods.

4 Experiments and Results

4.1 Dataset and Evaluation

The ALTA 2015 shared task is to identify all words in English texts from the perspective of the French language. Training data are provided, while labels of test data are not given. Since our system only focuses on limited similarity measurements, we believe a development set is not necessary. For each approach discussed, we use the training data to find the best threshold. Then, we test our system on the public testing data. If the results improve in both training and public testing, we submit our system.

The evaluation metric for this competition is F_1 score, which is commonly used in natural language processing and information retrieval tasks. Precision is the ratio of true positives (tp) to all predicted positives (tp+fp). Recall is the ratio of true positives (tp) to all actual positive samples (tp+fn).

$$P = \frac{tp}{tp + fp}, \; R = \frac{tp}{tp + fn}.$$

$$F_1 = 2\frac{P \cdot R}{P + R}$$

4.2 Experiment Results

We first compare bi-gram similarity and traditional edit distance similarity (Tables 5 and 6). SL, Max and Sqrt are all tested as normalization

Algorithm 1 Orthographic Edit Distance

```
1: function OrthEditDist($s$, $t$, $map$)
2:     $sl \leftarrow len(s)$
3:     $tl \leftarrow len(t)$
4:     for $i \leftarrow 0$ to $sl$ do
5:         $d[i][0] \leftarrow i$
6:     end for
7:     for $j \leftarrow 0$ to $tl$ do
8:         $d[0][j] \leftarrow j$
9:     end for
10:    for $i \leftarrow 0$ to $sl - 1$ do
11:        for $j \leftarrow 0$ to $tl - 1$ do
12:            $d[i+1][j+1] \leftarrow \min\{d[i+1][j]+1, d[i][j+1]+1, d[i+1][j+1]+1\}$
13:            for each orthographic pair $(s', t')$ in $map$ do
14:                $i' \leftarrow i - len(s')$
15:                $j' \leftarrow j - len(t')$
16:                if $i' \geq 0$ and $j' \geq 0$ then
17:                    continue
18:                end if
19:                if $s.substring(i', i+1) = s'$ and $t.substring(j', j+1) = t'$ then
20:                    $d[i+1][j+1] \leftarrow \min\{d[i+1][j+1], d[i'][j'] + \alpha\}$
21:                end if
22:            end for
23:        end for
24:    end for
25:    return $d[sl][tl]$
26: end function
```

L	Precision(%)	Recall(%)	F-1(%)
SL	72.49	79.52	75.84
Max	71.80	**80.96**	76.10
Sqrt	**75.23**	78.20	**76.68**

Table 6: Result of edit distance similarity on training dataset using different normalization methods.

L	Precision(%)	Recall(%)	F-1(%)
SL	77.56	75.15	76.34
Max	**75.48**	79.46	**77.42**
Sqrt	74.80	**79.82**	77.23

Table 7: Result of orthographic edit distance similarity on training dataset using different normalization methods.

factors for both approaches. Edit distance similarity constantly outperforms bi-gram similarity (around 0.5% to 1% higher). Orthographic edit distance similarity further improves the result by about 0.5%. Another trend is that Max and Sqrt normalization is better than SL, which only considers the length of source string. Max and Sqrt are competitive to some extent.

According to the previous experiment, we use orthographic edit distance similarity to measure the similarity of words. The maximum length of source word and target word is used as the normalization factor. Using the grid search algorithm, the threshold is set to 0.50. The final F_1 scores on public and private test data are 70.48% and 77.00%, both of which are at top place.

5 Conclusions

We used a translator and string similarity measures to approach the ALTA 2015 shared task, which was to detect cognates in English texts from the respect of French. By using our novel similarity method, orthographic edit distance similarity, our system produced top results in both public and private tests.

Acknowledgement

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

Chris Brew, David McKelvie, et al. 1996. Word-pair extraction for lexicography. In *Proceedings of the 2nd International Conference on New Methods in Language Processing*, pages 45–55. Citeseer.

Grzegorz Kondrak. 2005. N-gram similarity and distance. In *String processing and information retrieval*, pages 115–126. Springer.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

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

Klaus U Schulz and Stoyan Mihov. 2002. Fast string correction with levenshtein automata. *International Journal on Document Analysis and Recognition*, 5(1):67–85.

Esko Ukkonen. 1985. Algorithms for approximate string matching. *Information and control*, 64(1):100–118.