Automatic alignment of dictionaries for closely related languages

Elizaveta Kuzmenko, lizaku77@gmail.com

1. Introduction

Our paper presents research dedicated to automatic alignment of dictionaries for closely related languages. Automatic alignment means associating cognates in different languages with one another, so that there is a link between, for example, *сөз* ‘word’ in Kazakh and *сүз* ‘word’ in Tatar, *көз* ‘eye’ in Kazakh and *күз* ‘eye’ in Tatar, *қыз* ‘girl’ in Kazakh and *кыз* ‘girl’ in Tatar, and so on. It can be argued that this problem can be solved with the use of etymological database, but I explore this problem for under-resourced languages that do not have a lot of available linguistic resources.

The task of identifying cognates automatically using corpora is very useful for many applications because it allows maintaining connections between linguistic resources for different languages. For example, cognates identification can help in making queries into corpora of closely related languages simultaneously. This in turn will help linguists to find the same phenomena in several close languages by making fewer queries.

As the material for my research I take the data from two Turkic languages: Kazakh and Tatar. These languages are very close to each other and seem convenient for this research because we have linguistic data for them: linguistic corpora, dictionaries and lists of lemmas. The developed method of alignment then can be applied to other closely related languages.

I have set up the following tasks to solve during our research:

- carry out the identification of cognates using only corpora without additional linguistic resources such as translation databases and thesauri;
- apply to the task of cognates identification new methods which were not previously used in the field.

I performed the identification of cognates and alignment of dictionaries using string distance metrics, such as normalized Levenshtein distance, and distributional semantic models for the corpora of the languages in question.

The originality of our work lies in application of vector space models to this task, which can be considered a semantically oriented approach to cognate identification. As I will show in Section 2, previous attempts to solve this task used orthographic or phonetic similarity between words, whereas semantic similarity was not widely used.

I use corpora of Kazakh and Tatar languages. The data I have are texts in .prs format. From this format I extract word forms, lemmas and part-of speech tags, as well as frequency lists. The corpus of Kazakh is rather small: it contains 1,002,114 tokens. The corpus of Tatar is significantly larger: it contains 27,617,947 tokens.

The structure of the paper is as follows: Section 2 describes the work that was previously done in this field; in Section 3 my approaches to the problem are presented: using normalized Levenshtein distance and mapping together vector spaces for two languages. Section 4 concludes my work.

2. Related work

I will start with describing the previous as well as ongoing research connected to cognate identification and bilingual word recognition.

There are three widespread methods of identifying cognates:

- based on semantic similarity;
- based on phonetic similarity;
- based on orthographic similarity.

Semantic similarity, as it was said earlier, is not often used for this task; however, there were some attempts to employ external linguistic resources such as WordNet (Inkpen et al. 2005, Navlea et al. 2011). Such resources were used to identify words from one and the same semantic frame.

More often, phonetic similarity and orthographic similarity measures are used for cognates identification. They are combined in most cases because both scores are derived with the use of string similarity metrics, and sometimes it is even difficult to say what type of similarity was actually assessed. Actually, for many languages these two similarity measures are interchangeable because the pronunciation and phonetics are heavily dependent on the orthographic form of a word. This is the case for the languages that I study: both in Tatar and Kazakh orthography reflects the way the word is pronounced, with minor exceptions (for example, к in Tatar becomes velar if it stays before back vowels, nasal consonants are assimilated before stops, unstressed vowels are reduced or even can be elided, and several of these features can be found also in Kazakh).

The most common workflow when identifying cognates looks as follows: a considerable number of cognates is collected either manually or extracted with the help of some translation database. Then this set of cognates is analyzed in order to collect string similarity metrics such as XDice (Ciobanu & Dinu 2014), LCS (largest common subsequence) ratio and other less common metrics (Inkpen et al. 2005). After that, a machine learning model is trained on the basis of this set to recognize new words with similar characteristics, which are hypothesized to be also cognates. In other cases, methodology from computational biology is employed, such as Needleman-Wunsch algorithm, which allows alignment of pieces of genetic code.

All present studies were carried out using the data from widely used European languages, such as French, Spanish, Italian, sometimes German and English.

However, I do not consider this approach to be the best because it requires either a significant amount of manual preprocessing to be carried out or the use of additional linguistic resources, such as framebanks or translation databases (which are widely available for the languages studied). I aim to perform automatic identification of cognates and then alignment of dictionaries in a more unsupervised way, when the use of manual preprocessing and labelling is minimized, if not completely excluded.

Another point that I consider very important is that present approaches towards this task do not require a lot of linguistic knowledge, and this could be the reason why present results in cognate identification demonstrate precision and recall not higher than 90 percent. Maybe linguistic knowledge such as taking into consideration contexts in which words are met could add to the quality of cognate identification.

What I am trying to add is the use of distributional semantic models to enrich the methodology with the data about semantic similarity between words. I carry out this task with the help of word2vec tool[1] that efficiently builds vector spaces from the lemmatized or plain text. Overall, I can say that the goal of this work is to explore to what extent semantic similarity adds to the quality of cognate identification.

The assumption lying behind distributional semantics is that the meaning of a word is defined by its context (distributional hypothesis, see, for example (Lenci 2008)). Therefore, similar words can be found in similar contexts.

[1] https://code.google.com/p/word2vec/

What complicates the task is that I deal with words from different languages. But such vector models were successfully used for machine translation tasks, which also deal with correspondent words from different languages (Son et al. 2012). Linking vector spaces together has significantly improved the quality of statistical machine translations. Also it has been shown that vector spaces of different languages capture linguistic regularities (for example, proportions for vectors of words in different cases), which can be also used for cross-linguistic semantic tasks (Mikolov et al. 2013b).

3. Methodology

Preprocessing stage

As I said earlier, I carry out our task on the material of corpora for the Kazakh and Tatar languages. The Kazakh language is spoken in Kazakhstan (the majority of speakers live there), Mongolia, parts of China and several other Asian countries. It belongs to the Kipchak branch of the Turkic language family. The Tatar language is mostly spoken in the subregions of Russian Federation: the Republic of Tatarstan and the Republic of Bashkortostan. It belongs to the Kipchak-Bolgar branch of the Turkic language family.

The data for these corpora is stored in .prs format, which has the following format:

- each line presents linguistic features for a particular token in a tab-separated way;
- every morphological analysis occupies a separate line.

Overall, every word in the corpus can be assigned the following features:

- the number of a sentence in a text;
- the number of a word in a sentence;
- language (if different from the main language of a corpus);
- graphical representation of a word (for example, the use of capital letters);
- the number of morphological analyses for this word;
- the number of lemmata for the word;
- the number of the current analysis;
- word lemma;
- translation to English/Russian (depends on the corpus);
- part-of-Speech tag;
- morphological features not related to the inflectional paradigm of a word;
- morphological features related to the inflectional paradigm of a word;

- punctuation mark to the left of the word (if present);
- punctuation mark to the right of the word (if present);
- markers of the beginning or the end of a sentence.

From these files I extract the wordforms, the lemmas and the PoS-tags. In the Tatar languages the extracted lemmas for verb were stems, not full lemmas, due to the features of the morphological analyzer for Tatar. These stems are shorter by one letter then full verbs. Wordforms are glued together to recover the actual texts and are used to find the data about the contexts in which words are met. Also I form a list of lemmas from the corpus and enrich it with the information about the parts of speech.

This research can be divided into two major parts. In the first part I describe filtering out hypothetical cognates with the use of normalized Levenshtein distance. At that moment I receive a list of word correspondences from which some words are cognates and some are not. It can be said that this step exploits orthographic similarity of cognates like other common approaches.

In the second part I apply the knowledge extracted with the help of vector space models to the task. I try to determine cognates for words from one language by transforming their vectors into the vector space of another language. Then I find the word that has the most similar vector to the one found by the transformation function.

Computing Levenshtein distance

At this step I establish connections between lemmas of two languages by finding the word with the lowest Levenshtein distance and therefore the most similar word to the given one.

Levenshtein distance is a metric that shows how many operations on an input word I should perform to get an output word as the result. The allowed edit operations include insertion, deletion and substitution.

Normalized Levenshtein distance

I normalize the Levenshtein distance by the length of a word (the longest word from the pair is taken). That allows to take into consideration cases when, for example, the distance between a pair of words is 3, and the words are as long as 6 characters. Such words, obviously, are more close to each other than words of length 4 with the edit distance of 3. In the second case the words are completely different, and there are a lot of such words. As for the first case, such words are more appealing to our task and should be ranked higher in the list of possible cognates. The exact formula that I used is demonstrated in (1):

(1) $$Normalized\ LD = 1 - \frac{LD}{max(word1, word2)}$$

The resulting value lies between 1 and 0, where 1 means full resemblance (i.e. words are identical) and 0 means complete difference (the edit distance is equal or more than the length of one of the words).

Additionally, I filtered out words that were not of the same part of speech as the initial word from their pair. I did this because there is more possibility that cognates will be of the same part of speech. Also, this strategy is more suitable for our task of aligning dictionaries, where it is more useful to link words from the same part of speech, and finding cognates with other morphological characteristics is a secondary task.

When filtering out the words by their PoS-tags, I found out that the needed word that is a cognate has the highest value in the list of words with the same part of speech, so there is no need to take into account the whole list of possible cognates for the word, or even the top 3 words.

The pairs of possible cognates found by this method are shown in the Table 1 (examples are taken from the top of the computed list).

Table 1. Possible cognates produced by the Levenshtein distance.

Kazakh lemma	**Closest Tatar lemma**	**Normalized LD coefficient**
контраст (N)	контракт	0.875
түгелдеу (V)	*түгелдер*	0.875
былтырғы (ADJ)	былтыргы	0.875
майбалық (N)	майбалык	0.875
дырылдау (V)	дырылда	0.875
тасымал (N)	тастымал	0.875

Underlined in the table are the words that are indeed cognates, the words in italics have some relation between them (common root), but are not really cognates, and other words are completely unrelated.

This table is representative of the computed list in general in the sense that cognates can be found throughout the whole list (though there are more of them in the first half), and they are adjacent to non-cognates and other noisy words.

As we can see, I already have some results, and the next step should be filtering the list in such a way that non-cognates would be differentiated from cognates.

Another modification of the Levenshtein distance that I tried is assigning different weights to the operations of deletion, insertion and substitution. When I was looking at the cognates produced by the standard Levenshtein distance, I noticed that most of the cognates have one or two characters that differ in the two languages, and the rules of transition from one language to another are strict, so that particular characters in one language transform into particular characters in another language, and these transitions are not random. Therefore, we can modify the algorithm counting the Levenshtein distance, so that words with substituted letters, which is more common for cognates than insertion of a letter, would be considered closer to each other. Deletion of a letter, on the contrary, is seldom found within true cognate pairs, so the words that differ by a deleted letter would be considered farther from each other.

The result received with this modification of the Levenshtein distance was comparable to the one received with the help of the unmodified of the Levenshtein distance. A random excerpt from a list of possible cognates produced by the algorithm is presented in Table 2.

Table 2. Possible cognates produced by the modified Levenshtein distance.

Kazakh lemma	**Closest Tatar lemma**	**LD coefficient**
тармак (N)	тармак	0.92
майдан (N)	мәйдан	0.92
бадана (N)	багана	0.92
аманат (N)	яманат	0.92
зарлау (V)	зарлану	0.92
карбыз	карбыз	0.92
болғар	болгар	0.92

<u>мәжбүр</u>	<u>мәжбүр</u>	0.92
картоп	картон	0.92

Underlined are words that form good pairs of cognates, and other words are completely unrelated. We can see that a picture is slightly better than in the case of the standard Levenshtein distance, as more good cognate pairs are found on the top of the computed list. But still there are many words that differ by one character, but are completely unrelated. As I found out, from the first 50 words in the generated list, 32 could be considered cognates, judging by their translations and orthographic form, and other 18 words were random pairs.

Levenshtein distance normalized by the transition frequency

Another possible modification of the Levenshtein distance is taking into consideration the transition frequency for the characters in the already found pairs. In the previous version of the Levenshtein distance we found out that if we make substitution of characters more relevant for the metrics, than we have better results, but there are still unrelated words. If we take the possible pairs and estimate how often one character changes into another character, we can make conclusions about which transitions are more frequent among cognates. The words that differ by these characters are more likely to be cognates.

For example, I found out that the letter ‘і’ in Kazakh turns to the letter ‘е’ in Tatar more frequently than to ‘ы’ (118 transitions of ‘і’ to ‘е’ and 46 transitions of ‘і’ to ‘ы’). Therefore, we modify the Levenshtein distance in such a way that words with transition of ‘і’ to ‘е’ are considered to be closer to each other.

The standard algorithm for the Levenshtein distance was modified in such a way that the penalty for the replacement of a character was equal to 1 divided by the frequency of such transition in the list of pairs generated by the previous modification. For example, if ‘қ’ in Kazakh changes to ‘к’ in Tatar in the list 111 times, then the penalty for this transition when counting the Levenshtein distance is 1/111 = 0.01. On the contrary, if ‘ө’ in Kazakh changes to ‘г’ in Tatar just two times, the penalty for this transition is 0.5/ As for the operations of deletion and insertion, they were assigned penalties of 1.

The results produced by this modification of the Levenshtein distance are presented in Table 3.

Table 3. Possible cognates produced by the Levenshtein distance normalized

by transition frequency.

Kazakh lemma	**Closest Tatar lemma**	**LD coefficient**
бұзық (ADJ)	базык	0.9936
аумақ (N)	аулак	0.9876
летчик (N)	метчик	0.9861
қаратамақ (N)	карасакал	0.9845
марал (N)	карар	0.9844
қаһар (N)	каһәр	0.9838

As we can see, the results of this modification are less satisfying than in the case of the previous modification. The words with several transitions have received high scores and rised on the top of the list of possible cognates. A possible solution could be banning several transitions in one words or limiting possible transitions to several ones that are most frequent among good cognate pairs.

Finding corresponding vectors in other vector space

Methodology of distributional semantics and vector space modeling

This approach is based on the methodology described in (Mikolov et al. 2013) and general principles of distributional semantics. The core principle is that the meaning of the word is defined by its context. Therefore, words with close meaning should appear in similar context, and I can find them by simply counting their neighbors in a sentence.

When we are dealing with vector space models, we are dealing not with the words of some language, but with their representations. Representations can be of different kinds, but in vector space models that I use in this work a word is represented by a vector, i.e. a range of numbers that is unique for that word.

There are two ways to build a vector space model for a language. The first way is called a counting-based model. In this case, cooccurrences of one word with every other word are counted, and as the result I have a square matrix of a size X where X is a number of wordforms found in a corpus. This approach has a lot of drawbacks:

- it is computationally expensive;

- the resulting matrix is very sparse because a lot of words do not co-occur with each other;
- in order to work with one word you should work with the whole corpus.

The second way of building a vector space model is a more modern one. It is called a prediction-based model. It also outputs a vector for every word, but the dimensionality of vectors is smaller, it can equal to 1000, 500, or even 100, and this significantly lessens the amount of computational effort needed to work with the model. The model is trained in such a way that vectors are made of numbers that seem to be random, but the more similar the words are, the more similar their vectors are.

In our work I use a toolkit for vector space models called gensim[2]. I trained the models on the corpora of Kazakh and Tatar with the following characteristics:

- The algorithm of gathering contexts for a word was Skip-Gram. In this algorithm a sentence is read n-gram by n-gram in contrast to another algorithm available, where a sentence is treated like a “bag of words”.
- The dimensionality of vector was 500.
- Only the words that were encountered in the corpus more than 5 times were taken to the model.

I must say that distributional semantics in general and vector space models in particular were not previously tested for agglutinative languages, and it is not clear what the quality of the resulting models is.

The methodology itself is as follows:

- from the list of computed Levenshtein pairs I extracted 50 pairs of valid cognates, for examples see the Table 2:

Table 4. Examples of cognate pairs.

Kazakh word	**Tatar word**
алабұта	алабута
атбағар	атбагар
кәнізәк	кәнизәк

[2] http://radimrehurek.com/gensim/models/word2vec.html

майдан	мәйдан
дәріс	дәріс

- For every word in a pair I computed its vector in a corresponding model. For example, the word көз ‘eye’ in the corpus for Kazakh looks like that (its full dimensionality is 500):

3.87381278e-02 1.49710989e-02 1.99640933e-02 … -6.47256733e-04 2.12861840e-02

- After that, having 50 pairs of vectors, I trained linear regression on these data with the help of the toolkit scikit learn (Pedregosa et al. 2011). The idea is that the linear model finds correspondences between the pairs of vectors and can predict a vector from one model given a vector from another model.
- Then I modified a built-in function from the word2vec realization in gensim, so that given a vector it outputs a word whose vector is most similar to a given one.

All this algorithm allowed us is to map two vectors spaces of different languages together in order to find words in two languages whose meanings are very similar to each other.

Table of computed cognates

With the use of the described approach, I found for each word from the list of Kazakh lemma 5 closest words from Tatar. The selected results are demonstrated in the Table 3.

Table 5. Corresponding words in vector spaces of the two languages.

Kazakh word	**Corresponding Tatar words**
адамшылық ‘humanism’	шөкәтсез ‘miserable’, имән ‘oak’, яшьләр ‘the young’, көнләшү ‘be jealous’, Сиең
одағай ‘strange, extravagant’	художник ‘painter’, заман ‘time, epoch’, үрнәк ‘model, example’, йөкче ‘loader, porter’
өтініш ‘request, petition’	дуңгыз ‘pig’, фабрика ‘plant’, артык

	‘even, very’, буйга, самолет ‘aircraft’
құрсақ ‘belly, stomach’	кояш ‘sun’, самолет ‘aircraft’, күпчелек ‘folk, majority’, кысылып ‘through the teeth’, хайван ‘animal’

As I can see from the table, the results are not very inspiring. Sometimes there is some logic between the words from the Kazakh and the Tatar models, and sometimes they seem to be just a set of random words. But in all cases, I do not find cognates in the list of 5 semantically nearest words in the vector space of another language.

There can be many reasons for this. First, vector models for agglutinative languages were used for the first time, and their efficiency was not properly measured. If we take a list of frequent word and try to manually assess the quality of the output by the model, we can find that around 60% of words are semantically related to the given one. For example, if we take a word *сүз* ‘word’ from the model for Tatar, among its 10 nearest neighbors we can find such words as *хикәя* ‘story telling’, *сөйләү* ‘to speak’, *сөйләм* ‘(old) verb’, *фәлсәфәне* ‘philosophy’, *әкият* ‘fairytale’, but also *эзлеклелек* ‘sequence’, *таң* ‘dawn’, which are not connected in any way to the initial word. A proper research into the most suitable characteristics when applying distributional semantics to the languages in question needs to be done.

Second, it might be the case that the size of corpora used for building the models is not enough. Only 1 million tokens in the corpus of Kazakh was available to us, which is not enough to fully represent all the contexts in which the word can be met.

All in all, distributional semantics and vector space modeling might be of use when detecting cognates and finding words with similar meanings in different languages, but this approach need further research to be done.

4. Conclusion and future work

In our work I explored several methods of finding cognates in the corpora of closely related languages and aligning dictionaries for those languages automatically. In particular, I used two main methods:

- exploiting orthographic similarity between strings;
- exploiting semantic similarity between strings.

So far mostly orthographic similarity has been used for this task. I expected that considering semantic similarity will improve the results. I tried to employ semantic similarity with the use of vector space models, namely, word2vec implementation in gensim, where every word is represented as a vector in vector space.

However, this method, which was proven to work well for tracking semantic similarity, machine translation tasks, did not work out well for our task. The traditional method using the Levenshtein distance was found to be more effective, but still did not present the best results in the field because was used solely, without other supporting string similarity metrics.

Nevertheless, I am sure that semantic similarity can greatly improve the quality of detecting cognates. In future I am planning to investigate how semantic models in general and vector language models in particular can be used for this task. First of all, I should improve the quality of linguistic data used: augment the corpus of Kazakh with more texts, investigate the balanceness of corpora and the characteristics of vector models that will work better for the task.

Another possible way to improve the quality of cognates detection is to combine several approaches, for example, use orthographic and semantic similarity altogether. As it has been already stated for morphological disambiguation and many other linguistic tasks, combining different approaches works overall better than forcing one approach and trying to improve it more and more. So the possible solution can be making primary selection of cognates with the help of orthographic similarity and then improving the quality of aligned pairs with the use of semantic similarity.

References

Ciobanu, A. M., & Dinu, L. P. Automatic detection of cognates using orthographic alignment. In *Proceedings of The 52nd Annual Meeting of the Association for Computational Linguistics* (pp. 99-105).

Inkpen, D., Frunza, O., & Kondrak, G. (2005). Automatic identification of cognates and false friends in French and English. In *Proceedings of the International Conference Recent Advances in Natural Language Processing* (pp. 251-257).

Lenci, A. (2008). Distributional semantics in linguistic and cognitive research. *Italian journal of linguistics*, *20*(1), 1-31.

Mikolov, T., Le, Q. V., & Sutskever, I. (2013a). Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Mikolov, T., Yih, W. T., & Zweig, G. (2013b). Linguistic Regularities in Continuous Space Word Representations. In *HLT-NAACL* (pp. 746-751).

Navlea, M., & Todirascu, A. (2011). Using Cognates in a French-Romanian Lexical Alignment System: A Comparative Study. In *RANLP* (pp. 247-253).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825-2830.

Son, L. H., Allauzen, A., & Yvon, F. (2012). Continuous space translation models with neural networks. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 39-48). Association for Computational Linguistics.