Proceedings of 1st Workshop on Language Technologies for Historical and Ancient Languages, pages 10–16
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

Automatic Construction of Aramaic-Hebrew Translation Lexicon

Chaya Liebeskind, Shmuel Liebeskind
Computer Science Department, Jerusalem College of Technology, Lev Academic Center, Israel
liebchaya@gmail.com, israellieb@gmail.com

Abstract
Aramaic is an ancient Semitic language with a 3,000 year history. However, since the number of Aramaic speakers in the world has declined, Aramaic is in danger of extinction. In this paper, we suggest a methodology for automatic construction of Aramaic-Hebrew translation Lexicon. First, we generate an initial translation lexicon by a state-of-the-art word alignment translation model. Then, we filter the initial lexicon using string similarity measures of three types: similarity between terms in the target language, similarity between a source and a target term, and similarity between terms in the source language. In our experiments, we use a parallel corpora of Biblical Aramaic-Hebrew sentence pairs and evaluate various string similarity measures for each type of similarity. We illustrate the empirical benefit of our methodology and its effect on precision and F1. In particular, we demonstrate that our filtering method significantly exceeds a filtering approach based on the probability scores given by a state-of-the-art word alignment translation model.

Keywords: translation, lexicon, Aramaic, Hebrew, word alignment, string similarity

1. Introduction

A translation lexicon is a set of word pairs, where each pair contains one word from the source language and its translation equivalent (has the same meaning as, or can be used in a similar context to) from the target. Translation lexicons are an essential element of any statistical machine translation (MT) scheme. Previous work on MT has shown that, given sufficient parallel training data, highly accurate word translations can be learned automatically (Koehn et al., 2003; Chiang, 2007).
According to UNESCO, some 6,000-7,000 languages are spoken worldwide today. Approximately 97% are spoken by only 4% of the world population, while just 3% of the world speaks 96% of all the remaining languages. Most of those languages, mainly spoken by indigenous people, will alarmingly disappear. Thus, the worldwide preservation, revitalization and promotion of indigenous languages is urgent.
Aramaic is a member of the Afro-Asian language family's Semitic branch. Aramaic is an ancient language (closely related to both Hebrew and Arabic) with a 3,000 year history. Experts believe that Aramaic was main language from 539 BC to 70 AD in the Middle East and probably spoken by Jesus. However, as the number of speakers worldwide is declining, Aramaic is threatened by extinction.
Aramaic is the language of the Biblical books of Daniel and Ezra, and is the primary language of the Talmud (a key Jewish text) and the Zohar (a foundational work in the literature of Jewish mystical thought known as Kabbalah). To enable future scholars to understand and learn from these ancient texts in Aramaic, lexical resources, such as a dictionary, must be developed.
In this study, we present an algorithmic scheme for automatic construction of Hebrew-Aramaic translation lexicon. In particular, we propose and investigate a filtering process over an initial translation lexicon, generated by a state-of-the-art word alignment translation model. Our filtering method computes three types of string similarities, similarity between terms in the target language, similarity between a source and a target term, and similarity between terms in the source language. We examine five string similarity measures for the three types of similarity.
We demonstrate the empirical advantage of our scheme over a parallel Aramaic-Hebrew Biblical corpora and evaluate its impact on accuracy and F1. We show that our filtering method significantly outperforms a filtering approach based on the probability scores provided by the word alignment translation model. The remainder of this paper is organized as follows: we start with a description of word-based translation models that we utilize in our scheme and a brief summary on Aramaic natural language processing (NLP) (Section 2.). Then, we describe our Aramaic-Hebrew parallel corpora in Section 3.. Our main contribution of the algorithmic scheme is detailed in Section 4., followed by an evaluation in Section 5. and conclusions in Section 6..

2. Background

This section describes word-based translation models that we used in our experiments (Section 2.1.), followed by a brief introduction to the applications of NLP on our extinct language, Medium Aramaic, or "post-classical" Aramaic (Section 2.2.). We note that we also applied state-of-the-art neural MT algorithms. However, they did not perform well on our corpus, probably due to the limited amount of data.

2.1. Word-based Translation Models

Word alignment corresponds to word-based translation models (Brown et al., 1993), where the units of correspondence between sentences are individual words. Formally, we say that the objective of the word alignment task is to discover the word-to-word correspondences in a sentence pair ($F_1^J = f_1...f_J, E_1^I = e_1...e_I$) in which the source and target sentences contain I and J words, respectively.
An alignment A of the two correspondences is defined as (Och and Ney, 2003):

$$A \subseteq \{(j,i) : j = 1, ..., J; i = 0, ..., I\} \qquad (1)$$

in case that i = 0 in some (j, i) ∈ A, it represents that the source word j aligns to an "empty" target word e_0.

In statistical word alignment models, the probability of a source sentence given target sentence is written as:

$$P(f_1^J|e_1^i) = \sum_{a_1^J} P(f_1^J, a_1^J|e_1^i) \quad (2)$$

in which a_1^J denotes the alignment on the sentence pair. Several different parametric forms of $P(f_1^J, a_1^J|e_1^i) = p_\theta(f_1^J, a_1^J|e_1^i)$ have been proposed, and the parameters θ can be estimated using Maximum Likelihood Estimation (MLE) on a training corpus (Och and Ney, 2003).

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{s=1}^{S} \sum_{a} p_\theta(f_s, a|e_s) \quad (3)$$

The best alignment of the sentence pair, is called Viterbi alignment.

$$\hat{a}_1^j = \underset{a_1^J}{\operatorname{argmax}} p_\theta(f_1^J, a_1^J|e_1^i) \quad (4)$$

The IBM Models (Brown et al., 1993) are a sequence of word alignment models with increasing complexity, starting with lexical translation probabilities, adding models for reordering and word duplication. The IBM Models, along with the Hidden Markov Model (HMM) (Vogel et al., 1996), serve as the starting point for most current state-of-the-art statistical machine translation systems.
One of the serious drawbacks of the IBM models is that they create a one-to-many mapping. Their alignment function may return the same value for different input, but cannot return multiple values for one input (many-to-one). To resolve this and allow many-to-many mappings, various methods for performing a symmetrization of the IBM directed statistical alignment models are applied. Most of the symmetrization methods apply a heuristic postprocessing step that combines the alignments in both translation directions (source to target, target to source). If we intersect the two alignments, we get a high-precision alignment of high-confidence alignment points. If we take the union of the two alignments, we get a high-recall alignment with additional alignment points. In SMT (Och et al., 1999), a higher recall is more important (Och and Ney, 2000), so an alignment union would probably be chosen.
Och and Ney (Och and Ney, 2003) investigated the space between intersection and union with expansion heuristics that start with the intersection and add additional alignment points. They trained an alignment model in both translation directions and obtained two alignments a_1^J and b_1^I for each pair of sentences in the training corpus. Let $A_1 = \{(a_j, j)|a_j > 0\}$ and $A_2 = \{(i, b_i)|b_i > 0\}$ denote the sets of alignments in the two Viterbi alignments. They combined A_1 and A_2 into one alignment matrix A using the following steps:

1. Determine the intersection $A = A_1 \cap A_2$.
2. Extend the alignment A iteratively by adding alignments *(i, j)* occurring only in the alignment A_1 or in the alignment A_2:
    (a) if neither f_j nor e_i has an alignment in A, **or**
    (b) if both of the following conditions hold:
        i. The alignment *(i, j)* has a horizontal neighbor *(i - 1, j)*, *(i + 1, j)* or a vertical neighbor *(i, j - 1)*, *(i, j + 1)* that is already in *A*.
        ii. The set $A \cup \{(i, j)\}$ does not contain alignments with both horizontal and vertical neighbors.

In our experiments, we adopted this type of symmetrization methods, which are also known as *grow-diag-x* heuristics. Given this alignment method, it is quite straight-forward to estimate a maximum likelihood translation lexicon.

2.2. Aramaic NLP

Not much research has been done on Aramaic NLP. Some studies have used a corpus of ancient texts in mixed Hebrew and Aramaic language, the Responsa project. These studies discussed different tasks like abbreviation disambiguation (HaCohen-Kerner et al., 2010; HaCohen-Kerner et al., 2013), citations identification (HaCohen-Kerner et al., 2011; Koppel and Schweitzer, 2014), temporal data mining (Mughaz et al., 2017; Moghaz et al., 2019), and diachronic thesaurus construction (Zohar et al., 2013; Liebeskind et al., 2016; Liebeskind et al., 2019). Some of these studies have provided insights into the Aramaic language. However, since the main language of the Responsa project is Hebrew, these studies did not directly focus on Aramaic NLP.
Snyder and Barzilay (2008) presented a non-parametric model that jointly induces a segmentation and morpheme alignment from a multilingual corpus of short parallel phrases from the Hebrew Bible and translations (Targum Onkelos was used for the Aramaic (see Section 3.)). The model uses Dirichlet process prior for each language and for the cross-lingual links. Snyder and Barzilay (2008) applied their model to four languages: Arabic, Hebrew, Aramaic, and English. They showed that the joint model decreased error by up to 24% with respect to monolingual models. When used in languages of the same family, the model achieved better performance. However, since Snyder and Barzilay (2008) did not have gold standard segmentation for the English and Aramaic part of the data, they restrict their evaluation to Hebrew and Arabic.
An exception is the recent work of Porat et al. (2018) on identification of parallel passages in the Babylonian Talmud corpus. In contrast to the Responsa project, the Talmud main language is Aramaic. The method by Porat et al. (2018) allows for changes between the parallel passages on word level and on phrase level. On word level, they focused on the core of the words. First, the input corpus was used to compute the frequency of the Hebrew letters. Then, they identified for each word the two most rare letters and represented the word by these two letters (keep the order of two letters in the word). Since prefixes letters and matres lectionis are the most common letters in the language, The method by Porat et al. (2018) effectively eliminated most of them. They assumed that since they aimed to find sequences of matching two-letter codes, the number of false positives will be reduced later. On phrase level, they compared both n-grams of length 4 and non-contiguous n-grams

(termed skip-grams). They extracted all 4-word combinations for every 5-word sequence in the text, which could omit any of the last four words. Finally, to validate a given match, they clustered matching skip-grams by generating a two-dimensional graph. Each skip-grams match was plotted on one axis according to the base skip-gram starting word position, and on the other axis according to the corresponding skip-gram starting word position. Cases where several skip-grams match a cluster on the graph on a more or less diagonal line were considered valid. As the method by Porat et al. (2018) constructs its list of potential matches in a pre-processing step generated via a single pass, it is capable of processing text of any size in $O(N)$ time.

3. Parallel Corpus

Translation lexicon construction requires parallel data for learning. In a sentence-level parallel corpus, for every sentence in the source language there is a translated sentence in the target language. We used two Aramaic-Hebrew corpora:

1. **Targum Onkelos**, the Jewish Aramaic Targum, is an authentic translated text of the Pentateuch (Five Books of Moses), which is believed to have been written in the early 2^{nd} century CE. Its authorship is traditionally attributed to Onkelos, a well-known convert to Judaism in the Tannaic era (c. 35−120 CE). The Talmud story (Megillah 3a) tells that Targum Onkelos's content was first transferred to Moses at Mount Sinai by God, but later forgotten and recorded by Onkelos. Onkelos' Aramaic translation is a literal word-by-word translation, with very little additional material in the form of non-legalistic exegetical texts (usually where the original Hebrew is an idiom, a homonym, or a metapho). However, in cases where biblical passages are difficult, Onkelos aims at minimizing obscurities and ambiguities.

2. **Targum Jonathan**, the official eastern Aramaic translation to the Nevi'im (Prophets), the second main division of the Hebrew Bible. Its authorship is traditionally attributed to Jonathan ben Uzziel, a pupil of Hillel the Elder. The Talmud (Megillah 3a) states that "from the mouths of Haggai, Zechariah, and Malachi," suggesting Targum Jonathan was based on traditions derived from the last prophets. Its overall style is like Targum Onkelos, originated in the land of Israel and was accepted in Babylonia in the third century. Targum Jonathan was brought to the Diaspora by the Babylonian Academies.

4. Methodology

Translation lexicons usually contain thousands of entries, termed here *source terms*. Each entry holds a list of *target translated terms*, which has the same meaning as, or may be used in a similar context to the source term.

In this paper we assume that a sentence-level parallel corpus is given as input, and run an IBM method to extract a list of candidate target translated terms (termed *candidate translated terms*). Then, we focus on the process of filtering the candidate list and extracting a list of target translated terms for each source term.

Our methodology was performed on a Aramaic-Hebrew parallel corpus, but can be generically applied in other settings.

4.1. Algorithmic Scheme

We used the following algorithmic scheme for translation lexicon construction. Our input is a sentence-level parallel corpus. First, we extract an initial translation lexicon using an IBM word alignment algorithm. Next, to filter incorrect translations, for each term in the initial lexicon we retrieve all its translations and cluster them using some measure of string similarity. For example, the translation cluster of the Aramaic word גברין (men) is {אנשי, אנשים, איש} We consider clusters of more than one translation as valid and further examine clusters of length 1. We compare the similarity between the term and its single translation. A high similarity score indicates the correctness of the translation. For example, the Aramaic word אכלתון (eat) and its translation {אכלתם}. Finally, to check the validity of the remaining clusters (e.g. מדכר (ram)) and avoid losing cases like synonyms, we extract similar terms to the term that we are testing (דכרין) using some measure of string similarity and cluster the translations of all these terms ({אילם, האילים, איל, אילים, אילם}). If the cluster of the tested translation (איל) is contained in one of the extracted clusters, the translation is considered valid. The output is a filtered translation lexicon consisting of the translations which were judged valid by the algorithm.

The algorithm's pseudo code is described in Algorithm 1. String similarity measures are used (in four steps of the algorithm) to calculate three types of similarities; (1) similarity between terms in the target language (lines 4 and 13), (2) similarity between a source and a target term (line 7), and (3) similarity between terms in the source language (line 10).

4.2. String Similarity

Aramaic is a *resource-poor* language that lacks essential resources, such as part-of-speech taggers, necessary for computational linguistics. Thus, to calculate the similarity between two Aramaic words. we can not lemmatize them and compare their lemmas, but we need to apply a string similarity measure. Since Liebeskind et al. (2012) reported that available tools for Hebrew processing perform poorly on a diachronic corpus and our parallel corpora is of a similar genre, we also investigate string similarity measures for Hebrew. For word comparison in different languages (Aramaic and Hebrew), a string similarity measure is also required.

Table 1 lists the prior art string similarity measures considered in our work. Given two similar words with a different orthography, our goal is to find a measure which maximizes their similarity score or minimizes their distance score.

Although, in our corpora, the alphabet of both languages is the Hebrew alphabet. the letter distribution differ between the Aramaic and Hebrew. Figure 1 shows both the letters' frequency in each of the languages. ן and א are common

Algorithm 1: Methodology implementation

```
input  : A sentence-level parallel corpus
output: A translation lexicon
1  IntialLexicon ← IBM(parallel corpus);
2  foreach term t in IntialLexicon do
3      CandidateTransList ←
         GetCandTransList (t);
4      SimilarCandidateClusters ←
         GetSimCandCls(CandidateTransList);
       foreach cluster c in
       SimilarCandidateClusters do
5          if length(c)>1) then
6              add c to FilteredTransList
               break;
7          if IsTermCandSim(t,c) then
8              add c to FilteredTransList
9          else
10             SimilarTermList ←
                 GetSimilarTermList(t);
11             foreach term t2 in SimilarTermList do
12                 CandidateTransList + =
                     GetCandTransList (t2)
13             SimilarCandidateClusters ←
                 GetSimCandCls(CandidateTransList)
               foreach cluster simc in
               SimilarCandidateClusters do
14                 if length(simc) > 1  &  c ⊆ simc
                   then
15                     add c to FilteredTransList
16     add < t,FilteredTransList > to
       FilteredLexicon
```

Aramaic suffixes and ד is a common Aramaic prefix. Thus, they are more frequent in Aramaic than in Hebrew. On the contrary, ם is a common Hebrew suffix and ה and ש are common Hebrew prefixes, so they are more frequent in Hebrew.

5. Evaluation

5.1. String Similarity Evaluation

In our experiments, we investigated three types of similarities (see Section 4.1.), namely, Hebrew-Hebrew (HE-HE), Aramaic-Aramaic (AR-AR), and Aramaic-Hebrew (AR-HB). To evaluate the performance of the various string similarity measures presented in Section 4.2., we manually annotated word pairs of the three types. For each type, we annotated word pairs that were tested by the algorithm in the corresponding step. To avoid focusing on trivial cases that can easily be determined by all the measures, we only annotated pairs with at least two common letters, excluding matres lectionis.
Table 2 compares the performance of five string similarity measures for the three types of similarity by four commonly used measures: precision (P), recall (R), F1, and accuracy (Acc). For each configuration, we report the optimal threshold results. The word-level (Porat et al., 2018) measure was examined with the most two, three, and four rare letters, obtaining the best results with two lettered items for all the configurations.
In the sample of the annotated HE-HE pairs, there are 269 pairs, 187 positive (judged similar) and 82 negative (judged dissimilar). In the sample of the annotated AR-AR pairs, there are 559 pairs, 32 positive and 527 negative. In the sample of the annotated AR-HE pairs, there are 429 pairs, 131 positive and 298 negative. The gap between the number of positive and negative pairs of the corresponding configurations in the AR-AR sample explains the gap between the F1, which do not consider true negatives, and the accuracy, which does consider them.
The best results were obtained by the Jaro similarity measure, using different thresholds, for all the three types of pairs. The similarity thresholds were 0.67, 0.82, and 0.78 for HE-HE, AR-AR, and AR-HE, respectively.
To complete our investigation, we used the Hebrew part-of-speech tagger (Adler and Elhadad, 2006) to lemmatize the HE-HE pairs and compare their lemmas. We obtained recall, precision, F1, and accuracy of 0.4, 0.48, 0.44, and 0.29, respectively.
Next, we evaluated the performance of our algorithmic scheme. In all the reported results, for each similarity type, we used the best similarity measure with its optimal similarity threshold.

5.2. Algorithmic Scheme Evaluation

5.2.1. Evaluation Setting

The results reported in this paper were obtained from a sample of 108 randomly selected source terms from a list of 29,335 terms, generated by the state-of-the-art IBM model 4 using Giza++ (Och and Ney, 2003) open source toolkit[1]. Only source terms with more than one appearance in the corpora were selected. We manually annotated 287 source-target word pairs with an average of 2.66 word pairs per source term. Each source-target word pair was judged as either correct or incorrect translation.
We assessed our algorithmic scheme by evaluating its ability to filter the state-of-the-art IBM model 4 translation lexicon and increase its accuracy. Additionally, we compared our filtering process with a baseline filtering approach of deleting translations with low probability score. We used the probability (i.e. maximum likelihood) score that was assigned by the IBM model.
We used four evaluation measures: precision (P), relative-recall (RR), F1, and Accuracy (Acc). The scores were micro-averaged. Since we do not have any pre-defined translation lexicon, we evaluated the relative-recall. Our relative-recall considered the number of correctly translated source-target pairs from the output of state-of-the-art IBM model as the full set of translated pairs.
Table 3 presents the results of the baseline filtering approach which deletes translations with low probability score. We examined different probability thresholds. The best results were obtained with a threshold value of 0.1, which means almost no filtration. Once the filtering is more

[1] `http://www.statmt.org/moses/giza/GIZA++.html`

#	String Similarity Measure	Description
1	Levenshtein distance (Levenshtein, 1966)	Counts the minimum number of operations (removal, insertion, or substitution of a character) required to transform one string into another.
2	Hamming distance (Hamming, 1950)	Finds the total number of places one string is different from the other.
3	Jaccard similarity coefficient (Jaccard, 1901)	Counts the number of common characters and divides it by the total number of unique characters.
4	Jaro similarity (Jaro, 1989)	Highly scores strings with the same characters, but at a certain distance from each other, as long as the order of the matches is similar.
5	Word-level match (Porat et al., 2018)	Represents the words by their n most rare letters (keeps the order of the letters in the word) and requires an exact match.

Table 1: Prior art string similarity measures considered in our work

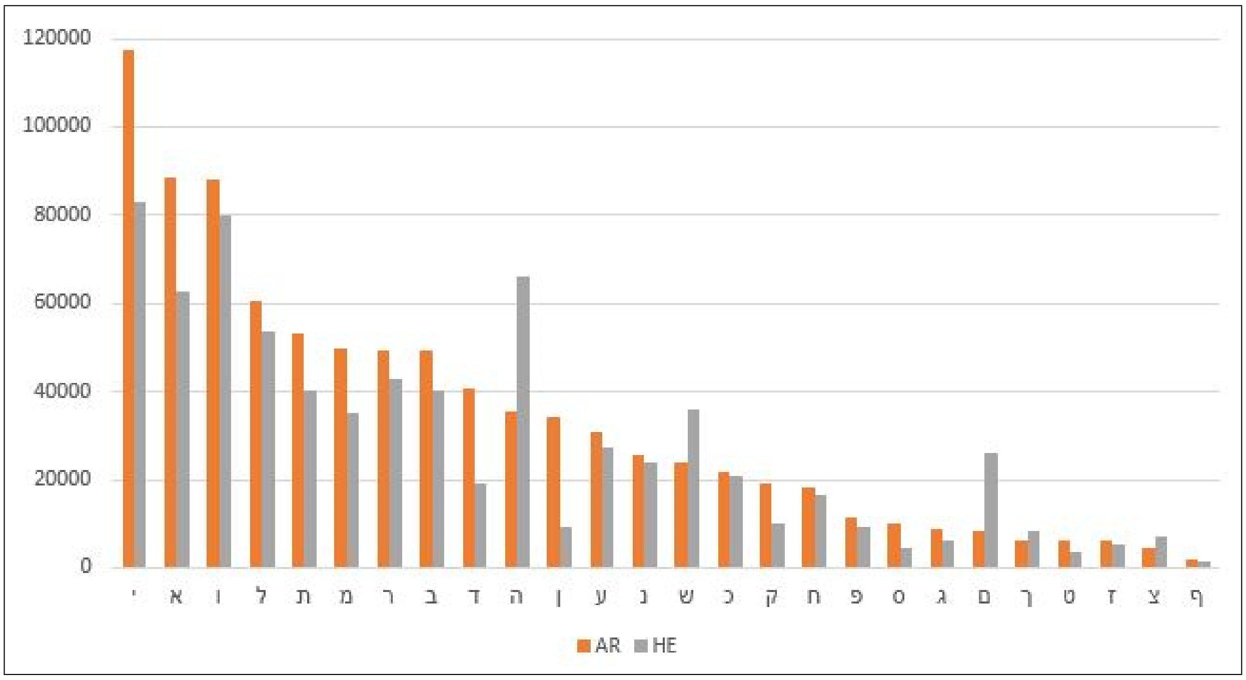

Figure 1: The frequency of the Aramaic and Hebrew letters.

Type	Measure	R	P	F1	Acc
HE-HE	Levenshtein	0.96	0.77	0.85	0.77
	Hamming	0.75	0.73	0.74	0.64
	Jaccard	0.8	0.9	0.85	0.8
	Jaro	0.89	0.83	**0.86**	**0.8**
	Word-level	0.65	0.9	0.75	0.71
AR-AR	Levenshtein	0.31	0.71	0.43	0.95
	Hamming	0.25	0.73	0.37	0.95
	Jaccard	0.41	0.5	0.45	0.94
	Jaro	0.41	0.72	**0.52**	**0.96**
	Word-level	0.19	0.26	0.22	0.92
AR-HE	Levenshtein	0.76	0.59	0.67	0.77
	Hamming	0.76	0.35	0.48	0.5
	Jaccard	0.65	0.72	0.69	0.82
	Jaro	0.69	0.8	**0.74**	**0.86**
	Word-level	0.49	0.79	0.61	0.81

Table 2: Performance of five string similarity measures for the three types of similarity

significant, there is very little precision increase and a dramatic recall drop. We concluded that the probability score is not sufficiently indicative to be used for filtering the initial lexicon and a different filtering scheme is required.

Threshold	RR	P	F1	Acc
0.1	0.85	0.798	0.823	0.711
0.2	0.758	0.789	0.773	0.648
0.3	0.634	0.8	0.708	0.585
0.4	0.533	0.823	0.647	0.54
0.5	0.498	0.819	0.619	0.516
0.6	0.33	0.852	0.476	0.425
0.7	0.291	0.846	0.433	0.397
0.8	0.273	0.838	0.412	0.383
0.9	0.251	0.826	0.385	0.366

Table 3: Baseline filtering approach with different probability thresholds

Table 4 compares the performance of our algorithmic scheme with that of the best baseline and the state-of-the-art IBM model 4. The state-of-the-art results corresponds to the baseline without any filtering. In other words, the IBM model classifies all the source-target pairs as positive. Therefore, its precision and accuracy are the same. Since we considered the correctly translated source-target pairs from its output as the full set of translated pairs, its relative-recall is 1.
Our algorithmic scheme increases both the F1 and the accuracy of the state-of-the-art IBM model by 5 points and 10 points, respectively. The baseline filtering does not improve the IBM model.

Method	RR	P	F1	Acc
Our Algorithmic Scheme	0.87	1	0.93	0.89
Baseline Filtering	0.85	0.8	0.82	0.7
State-of-the-art model	1	0.79	0.88	0.79

Table 4: Results Comparison

5.2.2. Error Analysis

We analyzed the classification errors of our algorithm. In Table 5, we present the classification confusion matrix. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class.

Predicted	True	False
Actual		
True	198	29
False	0	60

Table 5: The confusion matrix of our algorithm

All of the classification errors were due to incorrect classification of valid source-target pairs as invalid. In 34% of these incorrect classifications were cases where the translation appeared in a single morphology form. For example שמלתיו-כסותה (dress) and לאיש-לאנשא (to a man). The remainder of cases (66%) were classified incorrectly due to a low string similarity score. A low score was obtained in a few simple (ההר-טורא, בהר (mountain)), mediocre (אחיו-לאחוהי, אח (brother)) and more complex cases (החביאה-אטמרת, נחבאת (hide)). We note that the string similarity measure can be improved by matching terminal letters to regular letters as in the incorrectly classified example of כשדימה-כסדאי, כשדים (Chaldean (person)).

6. Conclusions and Future Work

We proposed a methodological algorithmic scheme to construct an Aramaic-Hebrew translation lexicon. First, by a state-of-the-art word alignment translation model, we generated an initial translation lexicon. We then filtered the initial lexicon using three types of string similarity measures. For each similarity type, we evaluated five string similarity measures. Our algorithmic scheme significantly increased both the accuracy of the F1 over the initial lexicon and a filtered lexicon based on word alignment probability scores. The scheme was investigated for Aramaic and Hebrew, but can be generically applied for other languages.
At some stage, during learning or in feature functions, all existing statistical machine translation (SMT) methods are using word alignments. Therefore, we plan to integrate our translation lexicon in a SMT scheme.

7. Bibliographical References

Adler, M. and Elhadad, M. (2006). An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 665–672. Association for Computational Linguistics.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Chiang, D. (2007). Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

HaCohen-Kerner, Y., Kass, A., and Peretz, A. (2010). Haads: A hebrew aramaic abbreviation disambiguation system. *Journal of the American Society for Information Science and Technology*, 61(9):1923–1932.

HaCohen-Kerner, Y., Schweitzer, N., and Mughaz, D. (2011). Automatically identifying citations in hebrew-aramaic documents. *Cybernetics and Systems: An International Journal*, 42(3):180–197.

HaCohen-Kerner, Y., Kass, A., and Peretz, A. (2013). Initialism disambiguation: Man versus machine. *Journal of the American Society for Information Science and Technology*, 64(10):2133–2148.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Koppel, M. and Schweitzer, N. (2014). Measuring direct and indirect authorial influence in historical corpora. *Journal of the Association for Information Science and Technology*, 65(10):2138–2144.

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

Liebeskind, C., Dagan, I., and Schler, J. (2012). Statistical thesaurus construction for a morphologically rich language. In * *SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 59–64.

Liebeskind, C., Dagan, I., and Schler, J. (2016). Semiautomatic construction of cross-period thesaurus. *Journal on Computing and Cultural Heritage (JOCCH)*, 9(4):22.

Liebeskind, C., Dagan, I., and Schler, J. (2019). An algorithmic scheme for statistical thesaurus construction in a morphologically rich language. *Applied Artificial Intelligence*, 33(6):483–496.

Moghaz, D., Hacohen-Kerner, Y., and Gabbay, D. (2019). Text mining for evaluating authors' birth and death years. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(1):7.

Mughaz, D., HaCohen-Kerner, Y., and Gabbay, D. (2017). Mining and using key-words and key-phrases to identify the era of an anonymous text. In *Transactions on Computational Collective Intelligence XXVI*, pages 119–143. Springer.

Och, F. J. and Ney, H. (2000). A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 1086–1090. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Och, F. J., Tillmann, C., and Ney, H. (1999). Improved alignment models for statistical machine translation. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Porat, E., Koppel, M., and Shmidman, A. (2018). Identification of parallel passages across a large hebrew/aramaic corpus. *Journal of Data Mining & Digital Humanities*.

Snyder, B. and Barzilay, R. (2008). Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, Ohio, June. Association for Computational Linguistics.

Vogel, S., Ney, H., and Tillmann, C. (1996). Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

Zohar, H., Liebeskind, C., Schler, J., and Dagan, I. (2013). Automatic thesaurus construction for cross generation corpus. *Journal on Computing and Cultural Heritage (JOCCH)*, 6(1):4.