AMR Alignment for Morphologically-rich and Pro-drop Languages

Elif Oral
NLP Research Group,
Faculty of Computer&Informatics,
Istanbul Technical University,
Istanbul, Turkey
oralk19@itu.edu.tr

Gülşen Eryiğit
Department of Artificial Intelligence &
Data Engineering,
Istanbul Technical University,
Istanbul, Turkey
gulsen.cebiroglu@itu.edu.tr

Abstract

Alignment between concepts in an abstract meaning representation (AMR) graph and the words within a sentence is one of the important stages of AMR parsing. Although there exist high performing AMR aligners for English, unfortunately, these are not well suited for many languages where many concepts appear from morpho-semantic elements. For the first time in the literature, this paper presents an AMR aligner tailored for morphologically-rich and pro-drop languages by experimenting on the Turkish language being a prominent example of this language group. Our aligner focuses on the meaning considering the rich Turkish morphology and aligns AMR concepts that emerge from morphemes using a tree traversal approach without additional resources or rules. We evaluate our aligner over a manually annotated gold data set. Our aligner outperforms the Turkish adaptations of the previously proposed aligners for English and Portuguese by an F1 score of 0.87 and provides a relative error reduction of up to 76%.

1 Introduction

AMR (Banarescu et al., 2013) is a semantic formalism that represents sentence meaning as directed graphs. The nodes in the graphs represent the *concepts* in the sentence, and the edges show the *relations* between the concepts. The purpose of AMR is to abstract sentence meaning from syntactic features. It gathers the semantic aspects of the sentence (semantic roles, time concepts, entity names, etc.) under a formalism and focuses on the sentence's meaning. Words that do not contribute to meaning and some syntactic features (tenses, passive voice, etc.) are not shown in AMR graphs.

With its increasing popularity in recent years, AMR has attracted the attention of many researchers (Žabokrtskỳ et al., 2020; Bos, 2016) and has been used in several applications such as text generation (Wang et al., 2020; Mager et al., 2020; Zhao et al., 2020; Fan and Gardent, 2020), text summarization (Dohare et al., 2017; Liu et al., 2018a; Liao et al., 2018), event extraction (Huang et al., 2016; Li et al., 2020). An important branch of this research is AMR parsing. Most parsing studies require an alignment between graph nodes and sentence concepts to create the training set for converting sentences to AMR graphs (Flanigan et al., 2014; Wang et al., 2015; Zhou et al., 2016). Many studies in the literature have reported that the alignment process greatly affects the parsing performance and has offered different solutions for the alignment process (Flanigan et al., 2014; Liu et al., 2018b; Pourdamghani et al., 2014; Anchiêta and Pardo, 2020). Lyu and Titov (2018) use alignments as latent variables during parsing, while Konstas et al. (2017); Zhang et al. (2019) could be given as an example study which does not require an alignment stage before parsing. However, since in these studies the learning process requires large amounts of sentence-AMR graph pairs, it is hard to apply them directly to a resource-poor languages.

The popular approaches in the literature for automatic AMR alignment aim to match concepts (either with fuzzy or semantic match) with the word lemmas with the help of a rule list. Although these approaches seem to be suitable for English, they do not perform well on languages with different characteristics (Anchiêta and Pardo, 2020). Morphologically-rich and pro-drop languages pose interesting challenges for AMR alignment as well as other NLP tasks. In these languages, many concepts appear from morpho-semantic elements rather than the entire word surface-form, yielding multiple concepts from a single word. In this paper, we introduce an AMR aligner for Turkish, a morphologically-rich and pro-drop language. Our alignment strategy handles concepts that emerge from the morphemes without the need for any extension of external resources or rules. Differing from the literature, with this approach, instead of

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Student Research Workshop, pages 143 - 152

looking for a match over the rule list, we use a two-stage strategy: we first map words to lexical concepts by using a similarity measure, and then navigate through the nodes over these matches, aligning the remaining concepts (i.e., abstract and morphology-based concepts) that do not have a word correspondence in the sentence. We evaluate our approach on manually annotated sentences randomly selected from IMST (Sulubacak et al., 2016) using the same evaluation methods from Flanigan et al. (2014). The results show that the proposed alignment strategy performs better for Turkish than the existing approaches originally proposed for English and Portuguese. The aligner will be available for researchers in GitHub [1].

The paper is organized as follows: Section 2 provides alignment fundamentals and briefly represents the related studies from the literature. Section 3 introduces the aligner and Section 4 gives the evaluation. Finally, 5 provides the conclusion.

2 Background and Related Work

In AMR parsing, although the parser takes sentences as input and produces AMR graphs, it is challenging to learn complete semantic representation from the sentences directly by using sentence-AMR graph pairs. Therefore, several parsing approaches need a word-concept alignment stage to know the semantic representation of words separately. One should note that concepts' names may differ substantially (e.g., due to inflections, derivations, or semantic closeness) from the related lexical words, and it is probable that they could not be mapped directly: for example, the word 'desirous' or 'desires' could be related to the concept name 'desire.01' in an AMR graph. This situation may be even harder in morphologically rich languages, where the character length of the inflectional affixes could be longer then the length of the lemma. Another example could be the word 'afraid' related to the concept 'fear-01'. An aligner is a tool that maps AMR concepts to the related words within the sentence. The outputs of the aligner are used as input data to train the AMR parsers.

JAMR (Flanigan et al., 2014) aligner, the first AMR aligner in the literature, is built on heuristic rules and greedy search. In this method, fuzzy matching between words and concepts is searched using heuristic rules. The aligner moves down from the first rule and looks for a fuzzy match based on the rule currently being processed. While some rules are applied to all nodes by traversing the entire graph for each rule, some are only applied to some specific nodes (e.g., entity names). The TAMR (Liu et al., 2018b) aligner is an extension of the method presented in JAMR with emphasis on meaning. The list of JAMR rules has been expanded with syntactic and semantic matching, where semantic and morpho-semantic matching are used together with fuzzy matching. The connection between verb-invoking nouns and their verb frames (e.g., example - exemplify) is provided by the morphological meaning database (Fellbaum et al., 2007). Pourdamghani et al. (2014) used syntax-based statistical machine translation with an unsupervised word alignment method in the alignment approach. During the alignment, they linearize AMR graphs with the IBM word alignment model (Brown et al., 1993) and map the nodes to English sentences. Anchiêta and Pardo (2020) presents an AMR aligner for Portuguese which is a morphologically rich language. The authors solve the word-concept alignment using the Word Mover's Distance (Kusner et al., 2015) and lexical lists for the alignment of the abstract concepts and entity names.

```
::snt The boy wants to be believed by the girl.
::alignments 1-2|0.0 2-3|0 5-6|0.1 8-9|0.1.0

(w / want-01                    0
   :ARG0 (b / boy)              0.0
   :ARG1 (b2 / believe-01       0.1
             :ARG0 (g / girl)   0.1.0
             :ARG1 b))
```

Figure 1: Alignment format of JAMR

The alignment format adopted in the literature is presented by JAMR, where alignment blocks are separated with white space (Figure 1). Each alignment block includes a word span and its graph fragment where a pipe sign ('|') separates them. Graph fragments consist of nodes represented with their position in the AMR graphs. A root node is located at position 0 ('want-01'); its children take 0.x where x represents the order of the children. For example, the first child of the root node takes 0.0 as a position indicator ('boy'); the second takes 0.1 ('believe-01').

[1] https://github.com/amr-turkish/turkish-amr-aligner

3 The Aligner

For Turkish, an alignment approach depending only on word-concept matching (Flanigan et al., 2014; Liu et al., 2018b) does not fully cover all concepts. On the other hand, unsupervised machine translation approaches (Pourdamghani et al., 2014) are not easily applicable due to representation issues (Oflazer and Durgar El-Kahlout, 2007) and the need for high-volume of parallel data. In Turkish, some of the correspondences are not present explicitly as a word, and are hidden inside the words as morphemes (e.g., personal markers, modality markers) due to its complex morphology and pro-drop nature. Consider the sentences in Figure 2 "Sana geleceğimi bilebilmene şaşırdım" (*I am surprised that you could know that I would be coming to you.*). The lemma of the words are 'sen' (*you*), 'gel'(*come*), 'bil' (*know*) and 'şaşır' (*be shocked*). The lexical concepts related to these may be aligned using fuzzy matching, however, the other concepts 'ben' (*I*) and mümkün.01 (*possible-01*)) deriving from their suffixes could not be matched. 'ben' is a dropped pronoun represented with a the first personal suffix (*-m*) that attach to verb lemma 'gel', 'mümkün.01' is originated from the modality marker (*-ebil*).

Figure 2 shows the aligned version of the sentence "Sana geleceğimi bilebilmene şaşırdım" (*I am surprised that you could know that I would be coming to you.*).

::snt Sana / geleceğ**imi** / bil**ebil**mene / şaşırdım
::eng to you / that I would be coming / that you could know / I am surprised
::alignments 0-1|0.1.1 1-2|0.1.1.0 2-3|0.1.0+0.1 3-4|0+0.0

```
(ş / şaşır.01                0
  :ARG0 (b / ben)            0.0
  :ARG1 (m / mümkün.01.      0.1
      :ARG1 (b2 / bil.01.    0.1.0
          :ARG0 (s / sen).   0.1.1
          :ARG1 (g / gel.01 0.1.1.0
              :ARG0 b
              :ARG4 s))))
```

Figure 2: AMR representation of the sentence "Sana geleceğimi bilebilmene şaşırdım" (*I am surprised that you could know that I would be coming to you*) and its alignment in JAMR format

To align such concepts, one alternative is to follow the literature and expand the rule list of JAMR (Flanigan et al., 2014) with the new rules to handle morphology based concepts. We believe this is not an option due to the following reasons: (i) There may be morphemes whose meaning can be changed according to the context. In order to align them, their meaning should be determined first and this needs semantic interpretation. Modality marker (*-meli*) is such an example and could carry out different meanings (i.e., 'should' or 'have to') depending on the context. (ii) There may be morphemes that invoke predicates, and the predicates invoked by the same morphemes can be different based on the nouns being attached. For example, when the very productive suffix *-CI* (with surface forms *ci, cı, çi, çı* under different vowel harmonies) attaches to nouns, it may mean 1) 'a person who sells' the item given in the noun lemma (e.g., 'simit**çi**' is the person who sells bagels where 'simit' is bagel), 2) 'a person who runs' the item given in the noun lemma (e.g., 'lokanta**cı**' is a person who runs a restaurant where 'lokanta' is restaurant) 3) 'a person who plays' (e.g., 'bas**cı**' is a person who plays bass guitar where 'bas' is bass guitar), and so forth. Covering all possible meanings with defining rules requires numerous rules. (iii) Construction of a solution on top of the morphemes (i.e., aligning morphemes using a predefined list) requires a preliminary morphological analysis stage. The aligner would become very dependent on the performance of the morphological analysis, and its errors propagate throughout the alignment. Considering these, we believe that a better approach should be proposed for the alignment of handling morphology-based concepts.

We propose an alignment strategy which relies on the word-concept similarity and tree traversal. Our aligner has two steps. In the first step, it builds a map where concepts are mapped to their word correspondence. This mapping is done by using similarity between pre-trained word embeddings for the words of the sentence and the node labels of the graph. The mapping does not necessarily include all concepts in this step: morphology derived and abstract concepts are left unmapped. The second step focuses on aligning all concepts. First, it starts aligning with concept-words pairs in the mapping obtained in the first step. Then it aligns the remaining concepts (i.e., morphology derived and abstract concepts) by traversing the AMR graph through the mapping. For each concept-word pair, the aligner visits neighbors of the concept by following the heuristically determined paths, and any

unaligned neighbors are simply added to the alignments for the word. Our aligner is detailed in the following subsections: Section 3.1 and 3.2.

3.1 Similarity Mapping

The similarity mapping aims to create lemma-concept pairs to be aligned in the next step. Our approach is similar to TAMR in which both syntactic and semantic similarities are used. However, we do not use morpho-semantic matching from TAMR despite Turkish being an morphologically rich language. Since Turkish is an agglutinative language, there is a direct link between the nouns invoking verbs and the verb frames. Therefore, semantic similarity can easily be used to match such nouns and the use of extra databases is not necessary.

The mapping is started with the semantic similarity calculation. A similarity score is calculated for each lemma-concept pair in the cross of lemma and concept set. We use Fasttext[2] (Grave et al., 2018) vectors, and empirically define threshold of 0.5 for the similarity score. Lemma-concept pairs with similarity scores above this limit are considered 'close'. The closest ones are matched with each other when they satisfy the condition that the closeness should be bi-directional. In other words, a mapping occurs when the closest concept of a lemma 'A' is B when B's closest pair is A (Function $mapping$ in Algorithm 2). It should be noted the aligner allows the lemma A to map more than one concept since there may be cases where A should have more than one concept as pair.

In some cases, word vectors fail to converge words having the same stem semantically; to handle them, we use syntactic similarity ($mFuzzy$) since their lemmas are the same[3]. After these two similarity matching processes, it is considered that remaining words that can not be mapped to any concept do not contribute to sentence meaning.

Similarity mapping seems straightforward; however, ellipsis makes the mapping difficult. An elliptical construction is the omission of one or more words that we call omitted words and their existence may be understood from the remaining words within the context. The AMR representation of such constructions varies across languages (Migueles-Abraira et al., 2018; Liu et al., 2019). Similar to (Liu et al., 2019), the omitted words are also restored and represented with concepts in Turkish AMR. This results in a situation that there may appear concepts whose correspondence words do not exist within the sentence. We call these concepts 'elliptic concepts' since they should align with the elided words. The elliptic concepts should be aligned with the words, but the aligned words may change according to the ellipsis type. Generally, we align elliptic concepts with the words that can help to understand the omitted words by semantic inference: these can be either the re-occurrences or the antecedents of the elided words. For the sake of simplicity, we name these words infer-words.

We gather the alignment of elliptic concepts under two categories: alignment with re-occurrences and alignment with antecedents. Similarity mapping of the first category is straightforward since re-occurrences can easily be matched with the elliptic concepts such as gapping ellipses. In the sentence "Herkes şeker (*verirdi*), o çikolata verirdi." (*Everybody would give chocolate*, s/he would give a candy.), 'verirdi' in parenthesis is omitted, but we can understand its existence by the last predicate (i.e., the infer-word). Its AMR graph has to have two 'ver.01' (give) frames since two different people perform different actions. We map both 'ver.01' concepts to 'verirdi'.

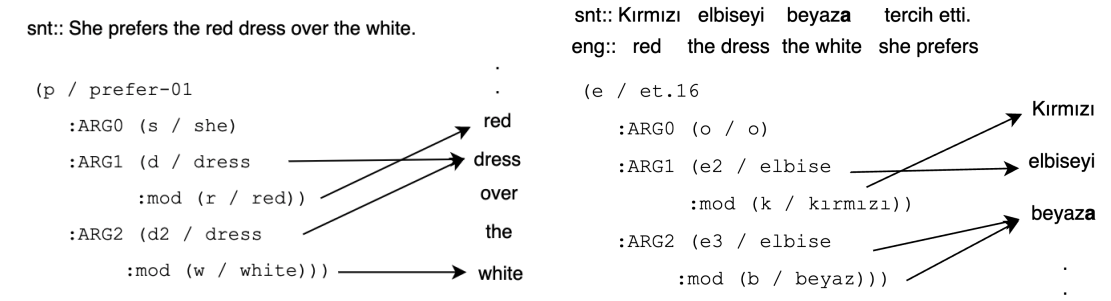
The latter category deserves more attention since the infer-words may cause ambiguity. Nominal ellipsis is such an example where there could exist syntactically similar words within the same sentence to the elliptic concrete concept, while the elliptic concept should actually be aligned to some other words (e.g., nominal adjectives) that derive the ellipsis instead of the syntactically similar one. This means that we need to match the concepts with the words that are not similar neither semantically nor syntactically, even if there exist completely identical words within the sentence.

Figure 3a provides such an example of the alignment of an English sentence ("S/he preferred the red dress over the white."). The elliptic concept (i.e., the second dress) is also derived from 'dress'. However, the morphologically-rich nature of Turkish poses extra challenges in such situations since the meaning carried by the 'dress' (first dress) will be provided by the suffix 'a' attached to the adjective (Figure 3b). This situation yields the need of mapping the elliptic concept to the nominal adjective.

To deal with this, we add a disambiguation (Funtion $disambiguation$ in Algorithm 2) step. Simi-

[2]https://fasttext.cc/docs/en/crawl-vectors.html

[3]We set threshold of 0.95 for the similarity score



(a) The alignment of nominal ellipsis in English. AMR graph on the left, the words of the actual sentence on the right. The elliptic concept 'd2' should be aligned with the word 'dress'.

(b) The alignment of nominal ellipsis in Turkish. AMR graph on the left, the words of the actual sentence on the right. The elliptic concept 'e3' should be aligned with "beyaza" (the lemma 'beyaz' (*white*) in dative form carrying the meaning of the dress).

Figure 3: The alignment of nominal ellipsis

larity mapping and disambiguation operate simultaneously. When there is more than one concept candidate paired with a single word within the sentence, a disambiguator is invoked to decide if a removal of a concept-word match is necessary: At this stage multiple mapping is allowed for the first category describe above (i.e., gapping ellipsis). For the second category (i.e., nominal ellipsis), the disambiguator reduces the possible multiple mappings into a selected one; i.e., selects one of the candidates.

The disambiguator searches for common syntactic structures (i.e., modifiers of the concept-lemma pair in focus) between the candidate concept and lemma. However, since we do not use any extra resources at this stage such as a dependency parser, we make a general assumption that the modifiers (e.g., adjectives describing a noun) would appear on the left side of a noun in the actual sentence word order. Although, most of the time, this assumption holds for English and several other languages, where modifiers frequently precede nouns, the direction may be changed if necessary for the language in focus.

First, the aligner calculates an overlap score between the 1-degree neighbors of each candidate concepts and the neighboring words in the 1-word window of the word in focus (i.e., the focus word to be mapped). Then, the candidate having a higher overlap score is matched with the word in focus. For example, possible mappings of the word in focus 'dress' in the actual sentence provided in Figure 3a are the two concepts d and d2. The overlap between the word in focus' neighbors is the word 'red', which eliminates the second possible mapping: the white dress. As one would notice, the introduced assumption does not have any effect in this example since the single overlapped word is enough for the elimination. However, when we look to the second example (Figure 3b) in the same figure, the 1-word window neighbor set of the word 'elbise' (*dress*) contain both the words 'red' and 'white' where our assumption helps to select the most possible candidate concept.

3.2 Alignment Algorithm

The alignment procedure (Algorithm 1) starts with similarity mapping ($simMap$) where word-concept pairs are determined as described in the previous section. Consider a set of words $W = \{w_1, w_2, ..., w_n\}$ in sentence S, where n is the number of words, the AMR graph is shown as $G = \{C, V\}$, where $C = \{c_1, c_2, ..., c_m\}$ is the set of concepts, and V is the relation set between these concepts. It should be noted that concept indexes and word indexes are not directly related to each other. As the output of similarity mapping, we get a list, each element of which is also a list (pl). This list pl contains $< w_j, c_i >$ pairs where j depicts the word-order index of the current word within the sentence. Our aligner processes each $< w_j, c_i >$ pair and first aligns c_i with w_j. Then it searches for c_i's one-edge away neighbors to find unmatched concepts during previous stage that need to be aligned with w_j. c_i is accepted as a central node and the aligner visits its neighbors. If a neighbor has a word pair, the aligner turns back to c_i. Otherwise the neighbor node is added to a list of visited concepts and the aligner moves to that node to search unmapped concepts in its neighborhood. This recursive process stops when there are no more mapped concepts in the neigh-

borhood and the aligner returns to c_i. Concepts added into the visited list during neighbor search are aligned with w_j. Then, the aligner moves to another concept-word pair and repeats the same steps. The alignment algorithm terminates when all pairs are processed.

Algorithm 1: Alignment Algorithm

Input: $S = List(w_1 \dots w_n), G = (C, V)$
Output: $Alignments$
$M \leftarrow simMap(S, G)$
$M \leftarrow sort(M)$
$T \leftarrow removeReent(G)$
$Alignments \leftarrow \emptyset$
for $j = 0$ ***to*** n **do**
  $pl_j \leftarrow M[j]$
  for $< w_j, c_i > \in pl_j$ **do**
    $Alignments[j] \cup \{c_i\}$
    $Alignments[j] \cup$ *getVisited*(c_i, T, M)
  end
end
$Alignments \leftarrow$ *postprocess*$(Alignments)$
Function *getVisited* (c, T, M)**:**
  $visited \leftarrow \emptyset$
  $n \leftarrow NeighInAllowedPath(c, T, M)$
  for $n_i, \in n$ **do**
    if $< \forall w, n_i > \notin M$ **then**
      $visited \cup$ *getVisited*(n_i, T, M)
  end
  return $visited$

Morphologicallly rich languages lead to frequent reification (i.e., conversion of a role into a concept (Banarescu et al., 2013)) situations in AMR. This results in nested concepts. Remember the example 'simitçi' (the person who sells bagels) from Section 3, which produces 3 concepts: 'person', 'sell.01', and 'bagel'. We use the above-explained recursive search for finding unmapped concepts within the nested relation chains.

At the beginning of the alignment procedure, we remove reentrancy[4] relations (function $removeReent$ in the Algorithm 1) and the graphs are transformed into trees. The reasons for this are that (i) we aim to align words whose alignments are graph fragments, and reentrancy connections appear on the linguistic phenomena such as co-reference, coordination, repetition, etc. (Blodgett and Schneider, 2021) rather than morphology-based ones where such graph fragments emerge. Therefore, we assume that the graph fragments do not include reentrancy connections. (ii) the majority of the reentrancy relations come from the personal suffixes whose concepts are mostly morphology originated. In figure 2, the concept 'ben' comes from the personal suffix *-Im* and can be aligned with 'geleceğ**imi**' or 'şaşırd**ım**', both alignments are correct. Since one of them is enough for the aligner to be used in concept generation as the first stage of parsing, we ignore the reentrancy connections during the alignment to be handled later during parsing.

Our aligner greedily searches neighbor nodes and the ordering of the concepts in the mapping list (M) is crucial for our aligner. The unmapped morphology-based concepts should be reached from their children nodes first since they tend to appear on top of the lexical concepts in the AMR graph. In order to ensure this, we add a sorting ($sort$) operation which moves the predicate concepts to the end of the mapping list to ensure that they are handled later than the leaf nodes. Moreover, we put constraints to force the aligner to visit neighbor nodes only in allowed path ($NeighInAllowedPath$) so that some nodes are reachable only via specified relations. These constraints guarantee the alignment of the abstract concepts of AMR. For instance, '*-quantity*' concepts are only reachable over the relations *:unit* and *:value*. The constraints are taken from JAMR rules responsible for alignments of abstract concepts.

Up to this stage, the aligner produces alignments for words. However, in order to produce alignments for word spans (e.g., named entities, reduplications, multi-word expressions), we need an additional stage to combine some words and their alignments. Therefore, we use a two stage post-processing step: The first stage focuses on the alignment of named entities: it unifies the alignment of consecutive words which were initially aligned to some concepts connected to the same 'name'[5] concept. In other words, consecutive words are merged into word spans, and their related concepts are also merged similarly for named entities.

[4] A single word in a sentence might be argument of more than one predicate. This is called reentrancy (Banarescu et al., 2013) in AMR.

[5] In AMR, the abstract 'name' concept is used for representing the named entities.

Algorithm 2: Similarity Mapping Algorithm ($simMap$)

Input: $S = List(w_1 \dots w_n)$, $G = (C, V)$, wv, t
Output: M
sim $\leftarrow n \times m$, M $\leftarrow \emptyset$
for $j = 0$ ***to*** n **do**
  $w_j \leftarrow S[j]$
  $v_{wj} \leftarrow wv(wj)$
  for $c_i \in C$ **do**
    $v_{c_i} \leftarrow wv(c_i)$
    if *CosSim*(v_{wj},v_{c_i}) $\geq t$ **then**
      sim[w_j][c_i]$\leftarrow$*CosSim*(v_{wj},v_{c_i})
    else if *mFuzzy*(v_{wj},$v_{c_i} \geq 0.95$ **then**
      sim[w_j][c_i]$\leftarrow$*mFuzzy*(v_{wj},v_{c_i})
  end
end
for $j = 0$ *to* n **do**
  $concepts \leftarrow$ *mapping*(sim,S,j,G)
  for each c_i *in* $concepts$ **do**
    M[j] $\cup\{< S[j], c_i >\}$
  end
end
Function *mapping* (sim,S,j,G)
  $w_j \leftarrow S$[j]
  $s \leftarrow$*max*(sim[w_j])
  for $c_l \in C$ **do**
    $w_k \leftarrow$*argmax*(sim[:,c_l])
    if $w_k = w_j$ **then**
      $cands \cup \{c_l\}$ if $s =$sim[w_k,c_l]
  end
  if $length(cands) \geq 2$ **then**
    $cands \leftarrow$
     disambugate($cands$,S,j,G)
  end
  return $cands$
Function *disambugate* ($candidates$,S,i,G)
  max $\leftarrow 0$, $v \leftarrow \emptyset$
  for each c *in* $candidates$ **do**
    if c *is verb frame* **then**
      return $candidates$
    else
      children $\leftarrow$ *getChildren*(c,G)
      prev $\leftarrow \{S[i-1]\}$
      next $\leftarrow \{S[i+1]\}$
      parents $\leftarrow$ *getParent*(c,G)
      overlap $\leftarrow$ { *prev* $\cap$ *children* } $\cup$
        { *next* $\cap$ *parents* }
      if $length(overlap) \geq$ max **then**
        $v \leftarrow$ candidate
        max $\leftarrow length(overlap)$
      end
    end
  end
  return { v }

The second stage focuses on multi-word expressions (i.e., idioms) and reduplications [6]. The previous stages can only align one word of such constructions, and the other words in the expression or reduplication remain unaligned. The post-processor checks each unaligned word within a sentence and creates word spans by combining them with their consecutive neighbours whose related concept name includes the unaligned word's lemma. For the languages which do not have these phenomena, this latter post processing steps could be omitted.

4 Experiments and Results

In order to evaluate the performance of our aligner, we randomly choose 100 sentences from a Turkish AMR corpus (Oral et al., 2022). and manually align the concepts of the AMR graphs with the words within sentences.[7]

Following the same evaluation method used in JAMR[8], we compare our aligner with TAMR (Liu et al., 2018b), JAMR (Flanigan et al., 2014) , and Portuguese-Aligner (shortly, PrAMR) (Anchiêta and Pardo, 2020). We adapt these aligners to Turkish. The predefined dictionaries such as months, conjunctions, etc., are replaced with their Turkish counterparts, and Fasttext is used in TAMR and PrAMR. We set similarity thresholds as 0.5 and 1.5 respectively. Since PrAMR uses lexical lists for named entities, we localized these as far as possible via a Portuguese to Turkish translator and adding additional Turkish gazetteers. Table 1 shows the results. Our aligner achieves an F1-score of 87% and outperforms the other aligners developed for English and Portuguese. Although TAMR and JAMR have a precision score relatively close to our aligner, they fall far behind the F1 score due to their low recalls. The recalls obviously show that the proposed methods for alignment fail to align around half of the concepts; these are the concepts derived from morphology. Furthermore, the alignment approach with Word Mover's Distance (PrAMR) has poorer performance than the fuzzy matching.

We design another experiment to evaluate our ap-

[6] Reduplication is the repetition of a word or part of a word (Göksel and Kerslake, 2004).

[7] The alignment gold set was annotated by one of the authors in two iterations. In the first iteration, the alignments were built from scratch. In the second iteration, the same annotator checked the correctness of the alignments, and the ones having alignment mistakes were corrected.

[8] https://github.com/jflanigan/jamr/blob/Semeval-2016/src/EvalSpans.scala

Output	P	R	F1
JAMR	0.73	0.48	0.58
TAMR	0.70	0.43	0.53
PrAMR	0.55	0.39	0.45
Ours	**0.89**	**0.84**	**0.87**

Table 1: The evaluation of our aligner

proach's effectiveness and investigate our aligner's alignment performance on different concept types. We evaluate our aligner on the sentence constituents concerning only the concept types in focus.

	P	R	F1
Elliptic Concepts	0.60	0.42	0.50
NEs	0.86	0.89	0.88
Abstract Concepts	0.90	0.82	0.86
Morphological Concepts	0.87	0.86	0.86

Table 2: Alignment performance of our aligner on different concept types

As shown in Table 2, our aligner's performance is in parallel to its overall score except elliptic concepts. Ellipsis is one of the most challenging parts of AMR alignment in Turkish. As a future work, we aim to improve the approach for this phenomenon.

We make a further error analysis to see the weaknesses of our aligner. One of our aligner's mistakes is the mismatch/unmatch of specific concepts. Since our alignment algorithm greedily searches unmapped concepts, any mistakes in the mapping phase result in the wrong alignments. Although our aligner uses the power of the pre-trained word embeddings, it fails to match the punctuation marks when they create concepts, especially the cases when they have a coordination role in the sentence: for example the comma mark is related to the concept 'and', and the colon mark is related to the concept 'de.01'(*say.01*), however these punctuation marks are not similar to the concept names neither semantically nor syntactically. The alignment of the light verbs is another unmatched case where our aligner fails. The Turkish Propbank (Şahin, 2016) represents them as frames of auxiliary verbs, which is how the AMR uses them too. Therefore, our aligner maps only the verb part due to semantic similarity; the first part of the verb is left unmatched. For example 'tercih et-' (*to prefer*) is represented with 'et.16' our aligner aligns only 'et' (*do*). Our aligner also shows poor performance on the alignment of the auxiliary verb 'ol'. This verb has 26 frames, including the widespread meanings 'have' (ol.04) and 'become' (ol.03). When there are multiple occurrences within the same sentence, the aligner does not have enough information to distinguish these frames. As a result, it may produce wrong mappings. An option to solve this ambiguity problem could be to integrate Propbank verb frames as an external resource in future works. One should note that this kind of additions would increase the alignment cost.

5 Conclusions and Feature Work

In this paper, we proposed an alignment approach for morphologically-rich and pro-drop languages and presented the first AMR aligner designed for Turkish which is prominent language of morphologically rich languages. Our aligner uses pre-trained word vectors and fuzzy matching for aligning concrete concepts. Furthermore, we present an algorithm for the alignment problem of concepts that emerged from the morphemes; this simple approach may be adopted to other morphologically rich and pro-drop languages with little effort. Our study reveals the challenging points in the Turkish alignment study, and we believe that our findings will accelerate the development of multilingual AMR parsing studies. As a future work, we plan to expand our study on the other morphologically rich and pro-drop languages (e.g., Portuguese).

References

Rafael Anchiêta and Thiago Pardo. 2020. Semantically inspired AMR alignment for the Portuguese language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1595–1600, Online. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Austin Blodgett and Nathan Schneider. 2021. Probabilistic, structure-aware algorithms for improved variety, accuracy, and coverage of amr alignments. *arXiv preprint arXiv:2106.06002*.

Johan Bos. 2016. Expressive power of abstract meaning representations. *Computational Linguistics*, 42(3):527–535.

Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Gözde Gül Şahin. 2016. Framing of verbs for Turkish propbank. *Turkish Computational Linguistics*, pages 3–9.

Shibhansh Dohare, Harish Karnick, and Vivek Gupta. 2017. Text summarization using Abstract Meaning Representation. *arXiv preprint arXiv:1706.01678*.

Angela Fan and Claire Gardent. 2020. Multilingual AMR-to-text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2889–2901, Online. Association for Computational Linguistics.

Christiane Fellbaum, Anne Osherson, and Peter E Clark. 2007. Putting semantics into wordnet's" morphosemantic" links. In *Language and technology conference*, pages 350–358. Springer.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.

Aslı Göksel and Celia Kerslake. 2004. *Turkish: A comprehensive grammar*. Routledge.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 258–268, Berlin, Germany. Association for Computational Linguistics.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. *arXiv preprint arXiv:1704.08381*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR.

Manling Li, Alireza Zareian, Qi Zeng, Spencer Whitehead, Di Lu, Heng Ji, and Shih-Fu Chang. 2020. Cross-media structured common space for multimedia event extraction. *arXiv preprint arXiv:2005.02472*.

Kexin Liao, Logan Lebanoff, and Fei Liu. 2018. Abstract meaning representation for multi-document summarization. *arXiv preprint arXiv:1806.05655*.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2018a. Toward abstractive summarization using semantic representations. *arXiv preprint arXiv:1805.10399*.

Yihuan Liu, Bin Li, Peiyi Yan, Li Song, and Weiguang Qu. 2019. Ellipsis in chinese amr corpus. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 92–99.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018b. An AMR aligner tuned by transition-based parser. *arXiv preprint arXiv:1810.03541*.

Chunchuan Lyu and Ivan Titov. 2018. AMR parsing as graph prediction with latent alignment. *arXiv preprint arXiv:1805.05286*.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. Gpt-too: A language-model-first approach for AMR-to-text generation. *arXiv preprint arXiv:2005.09123*.

Noelia Migueles-Abraira, Rodrigo Agerri, and Arantza Diaz de Ilarraza. 2018. Annotating abstract meaning representations for spanish. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Kemal Oflazer and İlknur Durgar El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 25–32, Prague, Czech Republic. Association for Computational Linguistics.

Elif Oral, Ali Acar, and Gülşen Eryiğit. 2022. Abstract meaning representation of Turkish. *Natural Language Engineering*.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning english strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429.

Umut Sulubacak, Gülşen Eryiğit, and Tuğba Pamay. 2016. IMST: A revisited Turkish Dependency Treebank. In *Proceedings of TurCLing 2016, the 1st International Conference on Turkic Computational Linguistics*, pages 1–6, Turkey. EGE UNIVERSITY PRESS.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. AMR-To-Text Generation with Graph Transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

Zdeněk Žabokrtskỳ, Daniel Zeman, and Magda Ševčíková. 2020. Sentence meaning representations across languages: what can we learn from existing frameworks? *Computational Linguistics*, 46(3):605–665.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. Broad-coverage semantic parsing as transduction. pages 3786–3798, Hong Kong, China. Association for Computational Linguistics.

Yanbin Zhao, Lu Chen, Zhi Chen, Ruisheng Cao, Su Zhu, and Kai Yu. 2020. Line graph enhanced AMR-to-text generation with mix-order graph attention networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 732–741, Online. Association for Computational Linguistics.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang Qu, Ran Li, and Yanhui Gu. 2016. Amr parsing with an incremental joint model. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 680–689.