Characters or Morphemes: How to Represent Words?

Ahmet Üstün **Murathan Kurfalı**
Cognitive Science Department
Informatics Institute
Middle East Technical University, Turkey
{ustun.ahmet,kurfali}@metu.edu.tr

Burcu Can
Department of Computer Engineering
Hacettepe University
Turkey
burcucan@cs.hacettepe.edu.tr

Abstract

In this paper, we investigate the effects of using subword information in representation learning. We argue that using syntactic subword units effects the quality of the word representations positively. We introduce a morpheme-based model and compare it against to word-based, character-based, and character n-gram level models. Our model takes a list of candidate segmentations of a word and learns the representation of the word based on different segmentations that are weighted by an attention mechanism. We performed experiments on Turkish as a morphologically rich language and English with a comparably poorer morphology. The results show that morpheme-based models are better at learning word representations of morphologically complex languages compared to character-based and character n-gram level models since the morphemes help to incorporate more syntactic knowledge in learning, that makes morpheme-based models better at syntactic tasks.

1 Introduction

The distributional hypothesis of Harris (1954) has been used to motivate work on vector space models to learn word representations. Deep learning models learn another kind of vector space model for building word representations, which shows superior performance in representing words.

Although deep neural networks have been very successful in representing words via such vectors, those models have not been very successful at estimating the representations of rare words since they do not appear often enough to allow us to collect reliable statistics about their context. Morphologically complex words are also rare by definition. Cao and Rei (2016) state that a word like *unbelievableness* does not exist in the first 17 million words of Wikipedia. Some methods have been proposed to deal with the sparsity issue in learning word representations. One approach is to utilize the subword information such as characters, character n-grams, or morphemes rather than learning distinct word representations without considering the inner structure of words.

Character-based models usually learn better word representations compared to word-based models since they capture the regularities inside the words so that it mitigates the sparsity in representation learning. However, those models learn the representations through the characters that do not correspond to a syntactic or semantic unit. In Turkish, two words can have similar word representations under a character-based model just because of their common suffixes. For example, character-based models such as (Bojanowski et al., 2017) generate similar word representations for words that have common character n-grams such as *kit**aplardan*** (from the books) and *kas**aplardan*** (from the butchers) (where *lar* and *dan* are suffixes, *kitap* and *kasap* are the roots) although the two words are semantically not related at all.

Another problem we observed for the character-based models is that such models estimate distant representations for words that are semantically related but involve different forms of the same morpheme so called allomorphs. This is one of the consequences of vowel harmony in some languages like Turkish. We observed this through several semantic similarity tasks performed on semantically similar but orthographically different words by using the word representations obtained from character n-gram level models such as fasttext (Bojanowski et al., 2017). For example, Turkish words such as *mavi**lilerinki*** (of the ones with the blue color) and ***sarılılarınki*** (of the ones with the yellow color) with allomorphs *li* and *lı*; *ler* and *lar*; *in* and *ın* are asserted to be distant from each other in regard to their word representations under a character n-gram level model such as fasttext (Bojanowski et al., 2017), although the two words are semantically similar and both referring to colors.

In this paper, we argue that learning word representations through morphemes rather than characters lead to more accurate word vectors especially in morphologically complex languages. Such character-based models are strongly affected by the orthographic commonness of words, that governs orthographically similar words to have similar word representations.

We introduce a model to learn morpheme and word representations especially for morphologically very complex words without using an external supervised morphological segmentation system. Instead, we use an unsupervised segmentation model to initialize our model with a list of candidate morphological segmentations of each word in the training data. We do not provide a single segmentation per word like others (Botha and Blunsom, 2014; Qiu et al., 2014), but instead we provide a list of potential segmentations of each word. Therefore, our model relaxes the requirement of an external segmentation system in morpheme-based representation learning. To our knowledge, this will be the first attempt in co-learning of morpheme representations and word representations in an unsupervised framework without assuming a single morphological segmentation per word.

Our model is mostly similar to that of Lazaridou et al. (2013) and Botha and Blunsom (2014) since we also aim to learn morpheme and word representations. Our model is akin to that of Pinter et al. (2017) from the training perspective since they infer the out-of-vocabulary word embeddings from pre-trained word embeddings. Here, we also try to mimic the word2vec (Mikolov et al., 2013) embeddings (i.e. that are the expected outputs of the model) to learn the rare word representations with a complex morphology.

Our model shows some architectural similarities to that of Cao and Rei (2016). Both models use the attention mechanism to up-weight the correct morphological segmentation of a word. However, their model is character-based and our model is morpheme-based where different segmentations of each word contribute to the resulting vector. It should be noted that our main concern is to investigate what character-based models cannot learn that the morpheme-based models learn. As for the experimental setting, we have chosen Turkish language that has a complex morphology and severe allomorphy.

The results show that a morpheme-based model is better at estimating word representations of morphologically complex words (with at least 2-3 suffixes) compared to other word-based and character-based models. We present experimental results on Turkish as an agglutinative language and English as a morphologically poor language.

2 Related Work

Classical word representation models such as word2vec (Mikolov et al., 2013) have been successful in learning word representations for frequent words. Since these classical models are based on collecting contextual information in a very large corpus, they estimate deficient word representations for rare words due to insufficient contextual information. This has a negative consequence in some natural language processing tasks that make use of the word representations.

One approach to overcome this deficiency in estimating rare word representations is to apply compositional methods. Each word comprises of different subword units, such as characters, character n-grams, or morphemes. Lazaridou et al. (2013) apply compositional methods by having the stem and affix representations in order to estimate the distributional representation of morphologically complex words. Bojanowski et al. (2017) introduce an extension to word2vec (Mikolov et al., 2013) by representing each word in terms of the vector representations of its n-grams, which was earlier applied by Schütze (1993) that learns the representations of fourgrams by applying singular value decomposition (SVD). Analogously, Alexandrescu and Kirchhoff (2006) represent each character n-gram with a vector representation and words are estimated by the summation of the subword representations. Their results show that compositional methods that are originally proposed for estimating the meaning of phrases can also be used for estimating the meaning of a word by combining the information coming from different subword units. Botha and Blunsom (2014) introduce

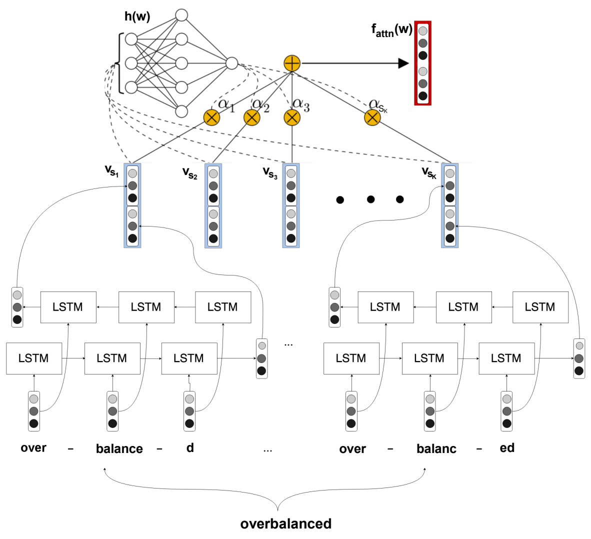

Figure 1: The neural network architecture of the morph2vec model.

a log-bilinear language model that integrates morphology with compositional methods, that is applied to translation task for morphologically rich languages.

Compositional models that use character-level features show that the representations of rare words can be estimated more accurately (in both semantic and syntactic tasks) than the word-based models since the character-level models share more features across different words that helps to mitigate sparsity. Cotterell and Schütze (2015) encode morphological tags within word embeddings by using a log-bilinear model, thereby leading morphologically similar words to have closer word representations in the embedding space. Luong et al. (2013) learn word representations based on morphemes that are obtained from an external morphological segmentation system. Collobert et al. (2011) enhance word vectors with some character-level features such as capitalization. Bhatia et al. (2016) incorporate morphological information as a prior distribution to improve word embeddings. They use Morfessor (Creutz and Lagus, 2002) as an external morphological segmentation system to extract the inner structure of words.

3 The morph2vec Model

In our morpheme-based model, a word is encoded by a sequence of morphemes. Each word w_{s_i} with a particular morphological segmentation s_i is represented by a list of morphemes $\mathbf{m} = \{m_0, m_1, \ldots, m_n\}$ as follows:

$$w_{s_i} = m_0 m_1 \ldots m_n$$

We assume that the correct morphological segmentation of a word is not known a priori by assuming a completely unsupervised learning model. We use an unsupervised neural segmentation algorithm (Üstün and Can, 2016) that generates a list of candidate segmentations for a given word (see Section 4 for the details).

Each distinct morpheme is defined by a column vector in a morpheme embedding matrix $W_m \in \mathbb{R}^{d_{morph} \times |M|}$ where d_{morph} is the vector dimension for the morphemes and M is the set of all pseudo morphemes.

Word representations are coupled with a particular morphological segmentation of each word. In other words, each segmentation of a single word has its own representation. Word representation for each particular segmentation is learned by a sequential function f that takes a sequence of morphemes and generates the word representation with a dimension of d_{word}. The word embedding that is to be estimated compositionally via its morphemes that belong to segmentation s_i is denoted by v_{s_i} and estimated by a function f as follows:

$$v_{s_i} = f(w_{s_i}) = f(v_{m_0}, v_{m_1}, \ldots, v_{m_n}) \qquad (1)$$

where v_{m_0} denotes the vector of m_0.

We use bidirectional LSTMs (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) to estimate a trainable function f in our neural network architecture that is illustrated in Figure 1. In the forward LSTMs, morphemes from the beginning till the end of the word are given sequentially, whereas in the backward LSTMs, morphemes from the end till the beginning of the word are given in the reverse order. Each output of Bi-LSTM which is the concatenation of the outputs of the forward and backward LSTMs represents a particular segmentation of a given word.

Therefore, we train the model with a list of potential segmentations of each word in training data. Since a word is represented by different morpheme sequences that refer to different segmentations of the same word, we use an attention model over these sequences that are learned by the Bi-LSTMs. Attention model learns a weight α_i for each segmentation, such that $\sum_i^{S_w} \alpha_i = 1$ where S_w denotes all potential morphological segmentations of w. The final word representation is the

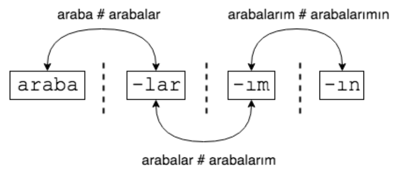

Figure 2: An illustration of generating potential segmentations for words in the training set.

weighted sum of the embeddings of all candidate segmentations:

$$f_{attn}(w) = \sum_i \alpha_i v_{s_i} \tag{2}$$

where v_{s_i} is the vector for segmentation s_i that is the output of a Bi-LSTM. The weight α_i is estimated as follows (Bahdanau et al., 2014):

$$\alpha_i = \frac{\exp(v^T \tanh(W \cdot v_{s_i}))}{\sum_j \exp(v^T \tanh(W \cdot v_{s_j}))} \tag{3}$$

Here, a feed-forward layer is used with a softmax function that is applied over the outputs of Bi-LSTMs. $W \cdot v_{s_i}$ denotes the corresponding column in the weight matrix of the feed-forward layer in the attention.

For training, we use the pre-trained word2vec (Mikolov et al., 2013) vectors in order to minimize the cost between the learned and pre-trained vectors with the following objective function:

$$J(\Theta) = \sum_{i=k}^{N} h(w_k) + \frac{\lambda}{2} \|\Theta\|_2^2 \tag{4}$$

where $h(w_k)$ is the cost for the kth word w_k in a training set of size N with a L2 regularization term on the model parameters θ. We use the cosine proximity loss between the learned and the pre-trained vector.

4 Neural Morphological Segmentation

Although it is possible to train the model by providing all the potential segmentations of each word, we utilize an unsupervised segmentation algorithm to make the model computationally more efficient by reducing the search space. The segmentation algorithm is based on the neural model by Üstün and Can (2016) that uses the semantic similarity between substrings of a word to detect the potential morpheme boundaries. This algorithm is based on the idea that the meaning of a word is preserved especially through inflection

s_0	araba - larım - ın
s_1	arabalar - ı - mın
s_2	arabalar - ım - ın
s_3	araba - lar - ı - mın
s_4	**araba - lar - ım - ın**

Table 1: Some candidate segmentations of the Turkish word *arabalarımın*. The bold one is the correct segmentation for this word.

Parent	Child	Sim.
araba	araba**-lar**	0.65
arabalar	araba**-lar-ım**	0.41
arabalarım	araba**-lar-ım-ın**	0.26

Table 2: The cosine similarities between the substrings (parent-child) of the Turkish word *arabalarımın* (of my cars). 0.25 is assigned for the cosine similarity threshold and only the splits above the threshold are listed.

and it benefits from the word representations to utilize this preservation. The parent-child relations such as (respect,respectful) are defined similar to that of Narasimhan et al. (2015).

The algorithm begins by generating all possible segmentations where there are at most K segments[1] (see Table 1). Then, the algorithm checks the semantic similarity at each split point (between the parent and its child) whether it is greater than a threshold[2]. If the condition is satisfied for all split points in a segmentation, the segmentation is added to the segmentations list that will be passed to a Bi-LSTM. Figure 2 illustrates an example for the segmentation algorithm on the Turkish word *arabalarımın (of my cars)*. # denotes a function that takes two words and returns true if the cosine similarity between two substrings is above the threshold value. The cosine similarities between the substrings of the word *arabalarımın* are given in Table 2.

Here we use the segmentation algorithm for mainly training purposes because the accuracy of the algorithm has a strong impact on generating word representations. Since all possible morphemes are not generated in training, if the segmentation algorithm generates an unknown morpheme in testing, the representation for that word involving the unknown suffix cannot be generated. In order to ensure that all morphemes have a rep-

[1]K is defined as 4 in all experiments.

[2]The threshold is assigned 0.25 in all experiments.

resentation, we use an external supervised segmentation system for only testing purposes. Another reason is that due to incorrect segmentations suggested by the unsupervised segmentation algorithm, two words (semantically related) involving the same set of suffixes cannot benefit from the syntactic similarity and therefore the representations of those words might diverge in testing.

5 Experiments

We performed several experiments to assess the quality of our morpheme and word embeddings. We did experiments on Turkish as a highly agglutinative language with a very complex morphology and English with a comparably poor morphology.

5.1 Experimental Setting

In all experiments, morpheme vectors have a dimension of $d_{morph} = 75$, while the forward and backward LSTMs have a dimension of $d_{LSTM} = 300$. Since the output of the Bi-LSTMs is the concatenation of the forward and backward LSTMs, the Bi-LSTM output has a dimensionality of $d_{biLSTM} = 600$. The output of the Bi-LSTMs is reduced to half after feeding the output through a feed-forward layer that results with a word vector dimension of $d_{word} = 300$. Our model is implemented in Keras, and publicly available[3].

For the pre-trained word vectors, we used the word vectors of dimension 300 that were obtained by training word2vec (Mikolov et al., 2013). For Turkish, we trained word2vec on Boun corpus (Sak et al., 2008) that contains 361 million word tokens. For English, we used the Google's pre-trained word2vec model[4] that was trained on 100 billion words with a vocabulary size of 3M. For training of our model, we used the most frequent 200K words from the pre-trained vocabularies to filter out the noise for both languages.

In order to compare the quality of our embeddings against the embeddings obtained from character n-gram level model fasttext (Bojanowski et al., 2017), we used the pre-trained word vectors trained on Wikipedia (Bojanowski et al., 2017) and we used the Google's pre-trained word vectors[5]. In order to compare our model with the character-based model by Cao and Rei (2016), we used Text8 corpus[6].

[3]http://nlp.cs.hacettepe.edu.tr/projects/morph2vec/

[4]https://code.google.com/archive/p/word2vec/

[5]https://code.google.com/archive/p/word2vec/

[6]Available at mattmahoney.net/dc/text8

Model	en	tr
word2vec (Mikolov et al., 2013)	0.69	0.483
fasttext (Bojanowski et al., 2017)	**0.71**	0.208
morph2vec	0.38	**0.529**

Table 3: The comparison of the Spearman correlation between the human judgments and the word similarities obtained by computing the cosine similarity between the learned word embeddings for English and Turkish.

Only for testing reasons, we used PC-KIMMO (Koskenniemi, 1984) for English and the two-level Turkish morphology (Akın and Akın, 2007) for Turkish in order to segment test sets to obtain the actual morphemes for generating word representations from the morpheme vectors that are learned in a fully unsupervised setting. Unsupervised segmentation system also could be used for the evaluation step, but we wanted to minimize the effect of incorrect segmentations to be able to evaluate the embeddings properly. Yet, we discuss the effect of the supervised vs unsupervised segmentations in Section 5.5.

We did only intrinsic evaluation with a set of experiments that assess the quality of the word and morpheme representations.

5.2 Evaluation of Word Representations: Word Similarity Results

In order to evaluate the quality of the word vectors, we did experiments on a list of word pairs. We computed the cosine similarity between the learned vectors of each word pair and compared the similarity scores against to human judgments.

We used the Set 2 in WordSim353 dataset (Finkelstein et al., 2001) for the semantic similarity experiments that already involves the human judgment scores from 1 to 10 for 200 English word pairs. Since there is no available word-pair list for Turkish, we prepared *WordSimTr*[7] that involves 138 word pairs and asked 15 human annotators to judge how similar two words are on a fixed scale from 1 to 10 where 1 shows a poor semantic similarity between the two words. Our Turkish word pair list involves two groups of words. The first group involves 81 semantically similar words that have at least two suffixes (possibly allomorphs). An example pair is *televizyonlarda* (on the televi-

[7]http://nlp.cs.hacettepe.edu.tr/projects/morph2vec/

Model	WordSim353	RW
char2vec (Cao and Rei, 2016)	0.345	0.284
morph2vec	**0.386**	**0.297**

Table 4: The comparison of the Spearman correlation between the human judgments and the word similarities obtained by computing the cosine similarity between the learned word embeddings for English on Text8 corpus.

sions) and *radyolarda* (on the radios) that have *lar* (for the plural) and *da* (for the locative case) with a semantically similar stem pair. The second group involves 57 semantically unrelated word pairs that are orthographically similar through their suffixes. An example word pair in this group is *kitaplardan* (from the books) and *kasaplardan* (from the butchers) with two suffixes *lar* (for the plural) and *dan* (for the ablative case) with semantically unrelated two stems *kitap* (the book) and *kasap* (the butcher). Some other example word pairs in the Turkish word pair list is given in Table 5. As seen on the table, our morpheme-based model is better at learning word representations with multiple suffixes.

The results are given in Table 3. English words mostly do not involve any suffixes, which hinders our model's performance. However, our model performs better than both fasttext (Bojanowski et al., 2017) and word2vec (Mikolov et al., 2013) on Turkish despite the highly agglutinative morphological structure of the language. It shows that our model learns better word representations for morphologically complex words, whereas words with no suffixes are not estimated as good as the complex ones.

We also compared our model against the character-based model char2vec (Cao and Rei, 2016). For this purpose, we trained our model on the same dataset and parameters as char2vec to be able to compare with their reported results. The dataset is called Text8 corpus and consists of the first 100mb of a cleaned-up dumb of Wikipedia in 2006. For the evaluation, we tested our word embeddings on Rare Words (RW) (Luong et al., 2013) and Wordsim353 (Finkelstein et al., 2001) datasets. The results are given in Table 4. Our results outperform char2vec (Cao and Rei, 2016) on both word similarity test sets. This shows that our model learns better word embeddings for both in-vocabulary and rare words compared to char2vec (Cao and Rei, 2016).

5.3 Evaluation of Word Representations: Analogy Results

We performed experiments for the analogy task in order to test whether the suffixes make a linear numerical change on the word vectors in the embedding space. The analogy experiments are usually performed for a triple of words such that A is to B so C is to ?, where A-B+C is expected to be equal to the questioned word. The analogy can be semantical such as *cat* is to *meow*, so *dog* is to *bark*, or syntactic such as *go* is to *gone*, so *have* is to *had*.

Here, we tested only the syntactic analogy on a list of word tuples since our focus is especially morphologically complex languages. For English, we used the syntactic relations section provided in the Google analogy dataset (Mikolov et al., 2013) that involves 10675 questions. Since there is no analogy dataset for Turkish, we prepared a Turkish analogy set *SynAnalogyTr*[8] with 206 syntactic questions that involves inflected word forms. The syntactic word tuples are judged by 40 human annotators in a scale from 1 to 10, where 1 shows a weak word analogy. Most words involve more than one suffix to test the morphological regularity in the analogy task.

The results are given in Table 6 and Table 7 for English and Turkish. The results show that our model outperforms both word2vec (Mikolov et al., 2013) and fasttext (Bojanowski et al., 2017) on both Turkish and English languages. Additionally, some examples to analogy results are given in Table 9 and the nearest neighbors of the Turkish word *kitap-lar-dan-mış* (it was from the books) are given in Table 8.

5.4 Evaluation of Morpheme Representations: Allomorph Results

In addition to the evaluation of the word vectors, we also evaluated the morpheme vectors that are the input embeddings to the neural network to be estimated during training. In order to evaluate how well our morpheme vectors represent the morphemes, we used the allomorphs. Allomorphs can be considered as *true synonyms* as they convey the same meaning with each other but with a different orthography.

[8]http://nlp.cs.hacettepe.edu.tr/projects/morph2vec/

Word Pair	Human Score	morph2vec	word2vec	fasttext
kitap-lar-dan / kasap-lar-dan (from the books) / (from the butchers)	0.12	0.07	0.19	0.55
kağıt-ta-ki-ler / bardak-ta-ki-ler (the ones on the paper) / (the ones in the glass)	0.22	0.12	OOW	0.64
şirket-ler-de / firma-lar-da (in the companies) / (in the firms)	0.87	0.82	0.46	0.76
kazanan-lar-dı / yenilen-ler-di (they were the winners) / (they were the defeated ones)	0.64	0.60	OOW	0.43

Table 5: Example Turkish word pairs and their similarities based on human judgements, morph2vec, word2vec (Mikolov et al., 2013), and fasttext (Bojanowski et al., 2017). - denotes the morpheme boundaries.

Model	Accuracy (%)
word2vec (Mikolov et al., 2013)	74.0
fasttext (Bojanowski et al., 2017)	74.9
morph2vec	**80.5**

Table 6: Analogy results on English Google syntactic analogy dataset.

Model	Accuracy (%)
word2vec (Mikolov et al., 2013)	16.0
fasttext (Bojanowski et al., 2017)	65.5
morph2vec	**71.3**

Table 7: Analogy results on Turkish syntactic analogy dataset *SynAnalogyTr*.

kitap-lar-dan-mış (it was from the books)	Cosine sim.
yazılmış (it was written)	0.669
hikaye-ler-den (from the stories)	0.667
kitap-lar (the books)	0.661
kitap-lar-dan (from the books)	0.639
kitap-lar-da (in the books)	0.635
roman-lar-dan (from the novels)	0.625

Table 8: Nearest neighbors of the word *kitap-lar-dan-mış* and the cosine similarity between the word and the neighbor that are obtained from morph2vec word vectors.

In Turkish, there is a common use of allomorphs due to the vowel and consonant harmony in the language. For example, the morpheme *dı* has got 8 allomorphs one of which is chosen depending on the last vowel and the consonant in the word, that are *di*, *du*, *dü*, *ti*, *tu*, *tü*, and *tı*. For example, *-ti* (the past tense of the third person singular) is chosen, for the verb *git-(mek)* (to go), whereas *du* is chosen for the verb *solu-(mak)* (to breathe).

We prepared a Turkish dataset that involves 108 morphemes[9] that are allomorphs of 33 unique morpheme types including tense and case markers as well as derivations. For the evaluation of allomorphs, we used the MAP metric that is often used in information retrieval tasks. For each allomorph set in the data, we calculated the MAP@k where k is the number of allomorphs for the given morpheme. If the allomorph of a morpheme exists in the k nearest neighbours, then it is regarded as correct, otherwise it is incorrect. We averaged the MAP@k scores for all allomorph sets. The results are given in Table 10. Our model can learn the morpheme representations better than fasttext (Bojanowski et al., 2017) since allomorphs in our model are closer to each other in the embedding space compared to fasttext. Some of the allomorphs obtained from our model and fasttext (Bojanowski et al., 2017) are given in Table 11. As seen on the table, our model can capture the allomorphs better than fasttext (Bojanowski et al., 2017). Additionally, all Turkish allomorphs learned by our model are given in Figure 3. As can be seen from the figure, the allomorphs fall into similar regions in the vector space. Apart from some infrequent morphemes, the rest has similar representations.

5.5 The Effect of Supervision

In our experiments, the model training is performed in a fully unsupervised setting in terms of

[9]http://nlp.cs.hacettepe.edu.tr/projects/morph2vec/

Word 1	Word 2	Word 3	Expected	morph2vec	word2vec	fasttext
gel-dim (I came)	gel-me-dim (I did not come)	duy-dum (I heard)	duy-ma-dım (I did not hear)	**0.80**	0.43	0.63
çöz-müş (she solved)	çöz-müş-tü (she had solved)	bul-muş (she found)	bul-muş-tu (she had found)	**0.73**	0.18	0.66
aç-mak (to open)	aç-ıl-mak (to be opened)	ört-mek (to cover)	ört-ün-mek (to cover himself)	**0.89**	0.36	0.52

Table 9: Example Turkish analogy questions and the cosine similarities between the expected words and the learned word representations obtained from morph2vec, word2vec and fasttext.

Model	MAP
fasttext (Bojanowski et al., 2017)	0.504
morph2vec	**0.618**

Table 10: MAP scores for the allomorph coverage in fasttext (Bojanowski et al., 2017) and the morph2vec.

morph	morph2vec	fasttext
iyor	ıyor yor uyor üyor	ıyor yor uyor üyor
mı	mu mi mü	mi mu **ıyor**
dı	tı di du tu dü ti **duk**	**dır** tı dü di **ın tır ı**
mış	müş muş	müş **yor dık**

Table 11: Some allomorphs of the given morpheme on the left that are found by our model and fasttext (Bojanowski et al., 2017). The bold font indicates the non-allomorphs for the given morpheme type.

Model	Spearman
Unsupervised (Üstün and Can, 2016)	0.517
Supervised (Akın and Akın, 2007)	**0.529**

Table 12: Word similarity Spearman correlation scores obtained when a supervised and unsupervised segmentation algorithm is used for the test sets.

the segmentation algorithm. However, we used supervised methods for segmenting test sets in evaluation to generate word representations from the actual morphemes that are learned in the training. We conducted another set of experiments with both supervised and unsupervised segmentation algorithms to show the effect of the segmentation algorithm used to generate the word embeddings in the word similarity test sets. Table 12 demonstrates the effect of the segmentation algorithm. Here, we employed the neural unsupervised model by Üstün and Can (2016) and the supervised segmentation system *Zemberek* (Akın and Akın, 2007).

The results show that we can generate better word embeddings when the morphemes are extracted by a supervised segmentation algorithm beforehand although the difference is not significant. Therefore the supervised segmentation algorithm used in testing can be replaced with an unsupervised segmentation algorithm.

However, it should be noted that when an unsupervised algorithm used, the possibility to come across with a segment that is not present in our model increases, hence we cannot generate the word embeddings for such words. Therefore, the results for the unsupervised setting is limited to only in-vocabulary words (i.e the words for which we can create a word embedding).

6 Conclusion and Future Work

Recent work shows that character level models learn more representative word embeddings for rare words (including morphologically complex words) compared to word level models, which is a sign that incorporating subword information improves the word representations. However, in this paper, we argued that morpheme-based representation models can learn better word embeddings (especially for the syntactic tasks) since they incorporate the syntactic and semantic information through the morphemes better compared to character level models. We pointed to the poor representation of allomorphs in complex words where the character-level models estimate a low word similarity between semantically similar words with different forms of the same morpheme, i.e. allomorphs. Moreover, we pointed to the character level models that assign a high word similar-

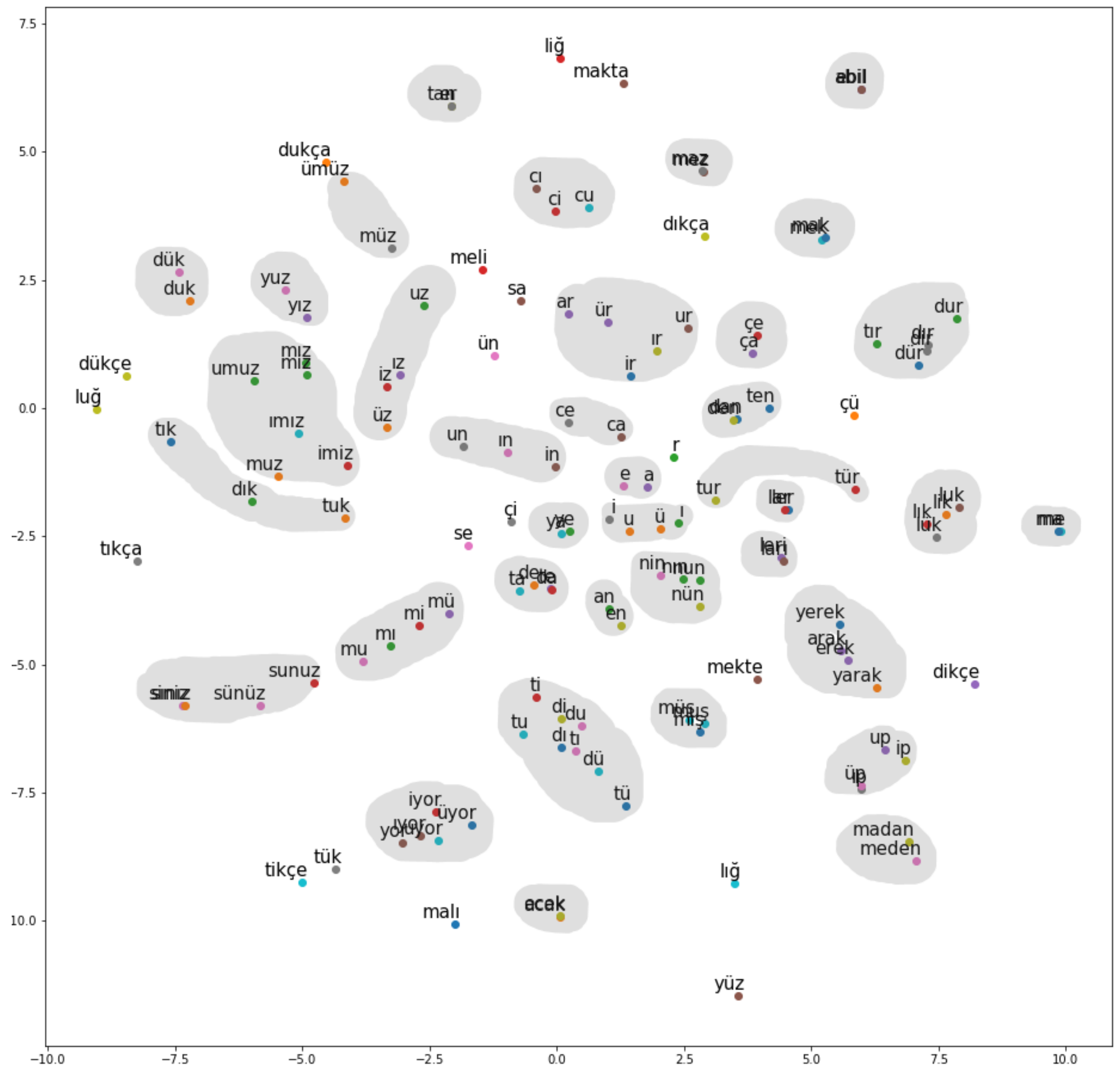

Figure 3: Turkish allomorph vectors learned by morph2vec. Some morphemes are blurry because of the overlapping of a few allomorphs.

ity to the words that are orthographically similar but semantically unrelated.

We introduce a morpheme-based representation model that learns word embeddings through the morphemes that are obtained from a list of morphological segmentations for each word. Therefore, our work introduces the idea of releasing the need for using an external morphological segmentation system in such representation learning models that are based on subword information. Our morpheme-based model *morph2vec* learns better word representations for morphologically complex words compared to the word-based model word2vec (Mikolov et al., 2013), character-based model char2vec (Cao and Rei, 2016), and the character n-gram level model fasttext (Bojanowski et al., 2017). Our results are also competitive for the English language.

We leave other languages and experiments such as morphological segmentation task for the future work. Another goal is to perform extrinsic evaluation on a different task such as part-of-speech tagging using the learned word embeddings.

Acknowledgments

This research was supported by TUBITAK (The Scientific and Technological Research Council of Turkey) grant number 115E464. We thank all the anonymous people for helping us to collect the human judgment scores for our word similarity and syntactic analogy datasets. We thank Suresh Manandhar for his valuable feedback.

References

Ahmet Afsin Akın and Mehmet Dündar Akın. 2007. Zemberek, an open source NLP framework for Turkic languages. *Structure* 10:1–5.

Andrei Alexandrescu and Katrin Kirchhoff. 2006. Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL-Short '06, pages 1–4.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 490–500.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. In *Transactions of the Association of Computational Linguistics*. TACL, pages 135–146.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. JMLR.org, ICML'14, pages II–1899–II–1907.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. *CoRR* abs/1606.02601.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word embeddings. In *Proceedings of the Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*. ACL'15, pages 1287–1292.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning - Volume 6*. Association for Computational Linguistics, Stroudsburg, USA, pages 21–30.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*. ACM, New York, NY, USA, WWW '01, pages 406–414.

Zellig Harris. 1954. *Distributional Structure*. Word.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kimmo Koskenniemi. 1984. A general computational model for word-form recognition and production. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '84, pages 178–181.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *ACL 2013 - 51st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*. pages 1517–1526.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 104–113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Karthik Narasimhan, Regina Barzilay, and Tommi S. Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *TACL* 3:157–167.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 102–112.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, pages 141–150.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2008. Turkish language resources: Morphological parser, morphological disambiguator and Web corpus. In *Advances in natural language processing*, Springer, pages 417–427.

Hinrich Schütze. 1993. Word space. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pages 895–902.

Ahmet Üstün and Burcu Can. 2016. Unsupervised morphological segmentation using neural word embeddings. In *International Conference on Statistical Language and Speech Processing*. Springer, pages 43–53.