A Joint Model for Word Embedding and Word Morphology

Kris Cao and Marek Rei
Computer Lab
University of Cambridge
United Kingdom
kc391@cam.ac.uk

Abstract

This paper presents a joint model for performing unsupervised morphological analysis on words, and learning a character-level composition function from morphemes to word embeddings. Our model splits individual words into segments, and weights each segment according to its ability to predict context words. Our morphological analysis is comparable to dedicated morphological analyzers at the task of morpheme boundary recovery, and also performs better than word-based embedding models at the task of syntactic analogy answering. Finally, we show that incorporating morphology explicitly into character-level models helps them produce embeddings for unseen words which correlate better with human judgments.

1 Introduction

Word embedding models associate each word in a corpus with a vector in a semantic space. These vectors can either be learnt to optimize performance in a downstream task (Bengio et al., 2003; Collobert et al., 2011) or learnt via the distributional hypothesis: words with similar contexts have similar meanings (Harris, 1954; Mikolov et al., 2013a). Current word embedding models treat words as atomic. However, words follow a power law distribution (Zipf, 1935), and word embedding models suffer from the problem of sparsity: a word like 'unbelievableness' does not appear at all in the first 17 million words of Wikipedia, even though it is derived from common morphemes. This leads to three problems:

1. word representations decline in quality for rarely observed words (Bullinaria and Levy, 2007).

2. word embedding models handle out-of-vocabulary words badly, typically as a single 'OOV' token.

3. the word distribution has a long tail, and many parameters are needed to capture all of the words in a corpus (for an embedding size of 300 with a vocabulary of 10k words, 3 million parameters are needed)

One approach to smooth word distributions is to operate on the smallest meaningful semantic unit, the morpheme (Lazaridou et al., 2013; Botha and Blunsom, 2014). However, previous work on the morpheme level has all used external morphological analyzers. These require a separate preprocessing step, and cannot be adapted to suit the problem at hand.

Another is to operate on the smallest orthographic unit, the character (Ling et al., 2015; Kim et al., 2016). However, the link between shape and meaning is often complicated (de Saussure, 1916), as alphabetic characters carry no inherent semantic meaning. To account for this, the model has to learn complicated dependencies between strings of characters to accurately capture word meaning. We hypothesize that explicitly introducing morphology into character-level models can help them learn morphological features, and hence word meaning.

In this paper, we introduce a word embedding model that jointly learns word morphology and word embeddings. To the best of our knowledge, this is the first word embedding model that learns morphology as part of the model. Our guiding intuition is that the words with the same stem have similar contexts. Thus, when considering word segments in terms of context-predictive power, the segment corresponding to the stem will have the most weight.

Our model 'reads' the word and outputs a sequence of word segments. We weight each segment, and then combine the segments to obtain the final word representation. These representations are trained to predict context words, as this has been shown to give word representations which capture word semantics well (Mikolov et al., 2013b). As the root morpheme has the most context-predictive power, we expect our model to assign high weight to this segment, thereby learning to separate root+affix structures.

One exciting feature of character-level models is their ability to represent open-vocabulary words. After training, they can predict a vector for any word, not just words that they have seen before. Our model has an advantage in that it can split unknown words into known and unknown components. Hence, it can potentially generalise better over seen morphemes and words and apply existing knowledge to new cases.

To evaluate our model, we evaluate its use as a morphological analyzer (§4.1), test how well it learns word semantics, including for unseen words (§4.2), and examine the structure of the embedding space (§4.3).

2 Related Work

While words are often treated as the fundamental unit of language, they are in fact themselves compositional. The smallest unit of semantics is the morpheme, while the smallest unit of orthography is the grapheme, or character. Both have been used as a method to go beyond word-level models.

2.1 Morphemic analysis and semantics

As word semantics is compositional, one might ask whether it is possible to learn morpheme representations, and compose them to obtain good word representations. Lazaridou et al. (2013) demonstrated precisely this: one can derive good representations of morphemes distributionally, and apply tools from compositional distributional semantics to obtain good word representations. Luong et al. (2013) also trained a morphological composition model based on recursive neural networks. Botha and Blunsom (2014) built a language model incorporating morphemes, and demonstrated improvements in language modelling and in machine translation. All of these approaches incorporated external morphological knowledge, either in the form of gold standard morphological analyses such as CELEX (Baayen et al., 1995) or an external morphological analyzer such as Morfessor (Creutz and Lagus, 2007).

Unsupervised morphology induction aims to decide whether two words are morphologically related or to generate a morphological analysis for a word (Goldwater et al., 2005; Goldsmith, 2001). While they may use semantic insights to perform the morphological analysis (Soricut and Ochs, 2015), they typically are not concerned with obtaining a semantic representation for morphemes, nor of the resulting word.

2.2 Character-level models

Another approach to go beyond words is based on on character-level neural network models. Both recurrent and convolutional architectures for deriving word representations from characters have been used, and results in downstream tasks such as language modelling and POS tagging have been promising, with reductions in word perplexity for language modelling and state-of-the-art English POS tagging accuracy (Ling et al., 2015; Kim et al., 2016). Ballesteros et al. (2015) train a character-level model for parsing. Zhang et al. (2015) do away with words completely, and train a convolutional neural network to do text classification directly from characters.

Excitingly, character-level models seem to capture morphological effects. Examining nearest neighbours of morphologically complex words in character-aware models often shows other words with the same morphology (Ling et al., 2015; Kim et al., 2016). Furthermore, morphosyntactic features such as capitalization and suffix information have long been used in tasks such as POS tagging (Xu et al., 2015; Toutanova et al., 2003). By explicitly modelling these features, one might expect good performance gains in many NLP tasks.

What is less clear is how well these models learn word semantics. Classical word embedding models seem to capture word semantics, and the nearest neighbours of a given word are typically semantically related words (Mikolov et al., 2013a; Mnih and Kavukcuoglu, 2013). In addition, the correlation between model word similarity scores and human similarity judgments is typically high (Levy et al., 2015). However, no previous work (to our knowledge) evaluates the similarity judgments

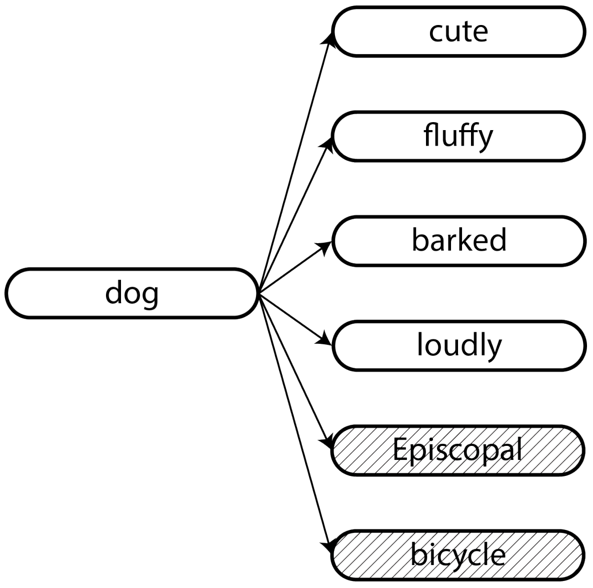

Figure 1: A graphical illustration of SGNS. The target vector for ‘dog’ is learned to have high inner product with the context vectors for words seen in the context of ‘dog’ (no shading), while having low inner product with random negatively sampled words (shaded)

of character-level models against human annotators.

3 The Char2Vec model

We hypothesize that by incorporating morphological knowledge directly into a character-level model, one can improve the ability of character-level models to learn compositional word semantics. In addition, we hypothesize that incorporating morphological knowledge helps structure the embedding space in such a way that affixation corresponds to a regular shift in the embedding space. We test both hypotheses directly in §4.2 and §4.3 respectively.

The starting point for our model is the skip-gram with negative sampling (SGNS) objective of Mikolov et al. (2013b). For a vocabulary V of size $|V|$ and embedding size N, SGNS learns two embedding tables $W, C \in \mathbb{R}^{N\times|V|}$, the target and context vectors. Every time a word w is seen in the corpus with a context word c, the tables are updated to maximize

$$\log \sigma(w \cdot c) + \sum_{i=1}^{k} \mathbb{E}_{\tilde{c}_i \sim P(w)}[\log \sigma(-w \cdot \tilde{c}_i)] \quad (1)$$

where $P(w)$ is a noise distribution from which we draw k negative samples. In the end, the target vector for a word w should have high inner product with context vectors for words with which it is typically seen, and low inner products with context vectors for words it is not typically seen with. Figure 1 illustrates this for a particular example. In Mikolov et al. (2013b), the noise distribution $P(w)$ is proportional to the unigram probability of a word raised to the 3/4th power (Mikolov et al., 2013b).

Our innovation is to replace W with a trainable function f that accepts a sequence of characters and returns a vector of length N (i.e. $f : A^{<\omega} \to \mathbb{R}^N$, where A is the alphabet we are considering and $A^{<\omega}$ denotes the finite length strings over the alphabet A). We still keep the table of context embeddings C, and our model objective is still to minimize

$$\log \sigma(f(w) \cdot c) + \sum_{i=1}^{k} \mathbb{E}_{\tilde{c}_i \sim P(w)}[\log \sigma(-f(w) \cdot \tilde{c}_i)] \quad (2)$$

where we now treat w as a sequence of characters. After training, f can be used to produce an embedding for any sequence of characters, even if it was not previously seen in training.

The process of calculating f on a word is illustrated in Figure 2. We first pad the word with beginning and end of word tokens, and then pass the characters of the word into a character lookup table. As the link between characters and morphemes is non-compositional and requires essentially memorizing a sequence of characters, we use LSTMs (Hochreiter and Schmidhuber, 1997) to encode the letters in the word, as they have been shown to capture non-local and non-linear dependencies. We run a forward and a backward LSTM over the character embeddings. The forward LSTM reads the beginning of word symbol, but not the end of word symbol, and the backward LSTM reads the end of word symbol but not the beginning of word symbol. This is necessary to align the resulting embeddings, so that the LSTM hidden states taken together correspond to a partition of the word into two without overlap.

The LSTMs output two sequences of vectors $h_0^f, \ldots, h_n^f$ and $h_n^b, \ldots, h_0^b$. We then concatenate the resulir ofting vectors, and pass them through a shared feed-forward layer to obtain a final sequence of vectors h_i. Each vector corresponds to two half-words: one half read by the forward LSTM, and the other by the backward LSTM.

We then learn an attention model over these hid-

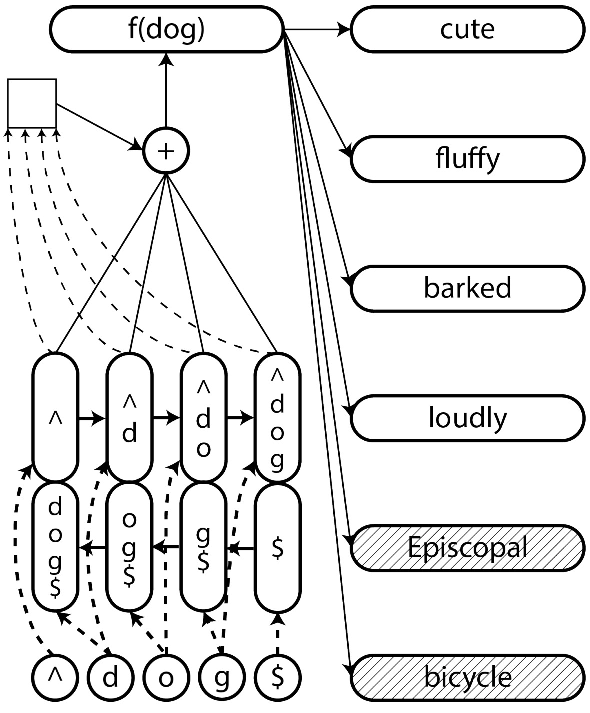

Figure 2: An illustration of Char2Vec. A bidirectional LSTM reads the word (start and end of word symbols represented by ^ and $ respectively), outputting a sequence of hidden states. These are then passed through a feed-forward layer (not shown), weighted by an attention model (the square box in the diagram) and summed to obtain the final word representation.

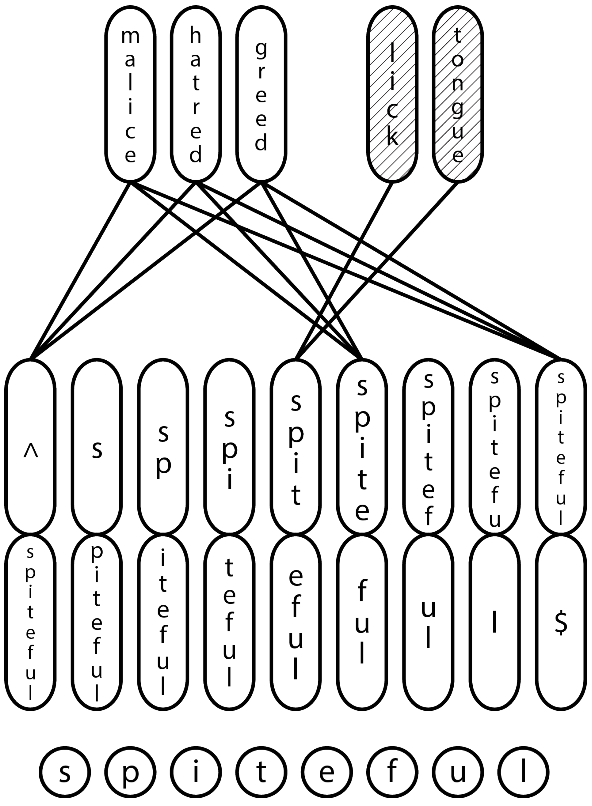

Figure 3: An illustration of the attention model (start and end of word symbols omitted). The root morpheme contributes the most to predicting the context, and is upweighted. In contrast, another potential split is inaccurate, and predicts the wrong context words. This is downweighted.

den states: given a hidden state h_i, we calculate a weight $\alpha_i = a(h_i)$ such that $\sum \alpha_i = 1$, and then calculate the resulting vector for the word w as $f(w) = \sum \alpha_i h_i$. Following Bahdanau et al. (2015), we calculate a as

$$a(h_i) = \frac{\exp(v^T \tanh(W h_i))}{\sum_j \exp(v^T \tanh(W h_j))} \quad (3)$$

i.e. a softmax over the hidden states.

3.1 Capturing morphology via attention

Previous work on bidirectional LSTM character-level models used both LSTMs to read the entire word (Ling et al., 2015; Ballesteros et al., 2015). This can lead to redundancy, as both LSTMs are used to capture the full word. In contrast, our model is capable of splitting the words and optimizing the two LSTMs for modelling different halves. This means one of the LSTMs can specialize on word prefixes and roots, while the other memorizes possible suffixes. In addition, when dealing with an unknown word, it can be split into known and unknown components. The model can then use the semantic knowledge it has learnt for a known component to predict a representation for the unknown word as a whole.

We hypothesize that the natural place to split words is on morpheme boundaries, as morphemes are the smallest unit of language which carry semantic meaning. We test the splitting capabilities of our model in §4.1.

4 Experiments

We evaluate our model on three tasks: morphological analysis (§4.1), semantic similarity (§4.2), and analogy retrieval (§4.3). We trained all of the models once, and then use the same trained model for all three tasks – we do not perform hyperparameter tuning to optimize performance on each task.

We trained our Char2Vec model on the Text8 corpus, consisting of the first 100MB of a 2006 cleaned-up dump of Wikipedia[1]. We only trained on words which appeared more than 5 times in our corpus. We used a context window size of 3 words either side of the target word, and took 11 negative samples per positive sample, using the same smoothed unigram distribution as `word2vec`. The model was trained for 3 epochs using the Adam optimizer (Kingma and Ba, 2015). All experiments were carried out using Keras (Chollet, 2015) and Theano (Bergstra et al., 2010; Bastien et al., 2012). We initialized the context lookup table using `word2vec`[2], and kept it fixed during training. [3] In all character-level models, the character embeddings have dimension $d_C = 64$, while the forward and backward LSTMs have dimension $d_{LSTM} = 256$. The concatenation of both therefore has dimensionality $d = 512$. The concatenated LSTM hidden states are then compressed down to $d_{word} = 256$ by a feed-forward layer.

As baselines, we trained a SGNS model on the same dataset with the same parameters. To test how much the attention model helps the character-level model to generalize, we also trained the Char2Vec model without the attention layer, but with the same parameters. In this model, the word embeddings are just the concatenation of the final forward and backward states, passed through a feedforward layer. We refer to this model as C2V-NO-ATT. We also constructed count-based vectors using SVD on PPMI-weighted co-occurence counts, with a window size of 3. We kept the top 256 principal components in the SVD decomposition, to obtain embeddings with the same size as our other models.

4.1 Morphological awareness

The main innovation of our Char2Vec model compared to existing recurrent character-level models is the capability to split words and model each half independently. Here we test whether our model segmentations correspond to gold-standard morphological analyses.

We obtained morphological analyses for all the words in our training vocabulary which were in the English Lexicon Project (Balota et al., 2007). We then converted these into surface-level segmentations using heuristic affix-matching, and used this as a gold-standard morphemic analysis. We ended up with 14682 words, of which 7867 have at least two morphemes and 1138 have at least three.

Evaluating morphological segmentation is a long-debated issue (Cotterell et al., 2016). Traditional hard morphological analyzers are normally evaluated on border F_1 – that is, how many morpheme borders are recovered. However, our model does not actually posit any hard morpheme borders. Instead, it just associates each character boundary with a weight. Therefore, we treat the problem of recovering intra-word morpheme boundaries as a ranking problem. We rank each inter-character boundary of a word according to our model weights, and then evaluate whether our model ranks morpheme boundaries above non-morpheme boundaries.

We use mean average precision (MAP) as our evaluation metric. We first calculate precision at N for each word, until all the gold standard morpheme boundaries have been recovered. Then, we average over N to obtain the average precision (AP) for that word. We then calculate the mean of the APs across all words to obtain the MAP for the model.

We report results of a random baseline as a point of comparison, which randomly places morpheme boundaries inside the word. We also report the results of the Porter stemmer[4], where we place a morpheme boundary at the end of the stem, then randomly thereafter.

Finally, we trained Morfessor 2.0[5] (Creutz and Lagus, 2007) on our corpus, using an initial random split value of 0.9, and stopping training when the difference in loss between successive epochs is less than 0.1% of the total loss. While Morfessor is no longer state-of-the-art in morpheme recovery (see, e.g. Narasimhan et al. (2015) for more recent work), it has previously been as a component in pipelines to build compositional word representations (Luong et al., 2013; Botha and Blunsom, 2014). We then used our trained Morfessor model to predict morpheme boundaries[6], and randomly permuted the predicted morpheme boundaries and ranked them ahead of randomly permuted non-morpheme boundaries to calculate MAP.

[1]available at `mattmahoney.net/dc/text8`

[2]We use the Gensim implementation: `https://radimrehurek.com/gensim/`

[3]We experimented with updating the initialized context lookup tables, and with randomly initialized context lookups, but found they were influenced too much by orthographic similarity from the character encoder.

[4]We used the NLTK implementation

[5]We used the Python implementation

[6]We found Morfessor to be quite conservative by default in its segmentations. The 2nd ranked segmentation gave better MAPs, which are the results we describe.

Model	All word MAP	Rich-morphology MAP
Random	0.233	0.261
Porter Stemmer	**0.705**	0.446
Morfessor	0.631	0.500
Char2Vec	0.593	**0.586**

Table 1: Results at retrieving intra-word morpheme boundaries.

Word	Model analysis	Gold-standard analysis
carrying	carry |ing	carry |ing
leninism	lenin |ism	lenin |ism
lesbianism	lesbia |nism	lesbian |ism
buses	buse |s	bus |es
government	gove |rnment	govern |ment
unrepentant	un |repent |ant	un |repent |ant
weaknesses	weak |nes |ses	weak |ness |es

Table 2: Morphological analyses for sample words from the corpus. We take the top N model predictions as the split points, where N is the number of gold-standard morphemes in the word.

As the test set is dominated by words with simple morphology, we also extracted all the morphologically rich words with 3 or more morphemes, and created a separate evaluation on this subsection. We report the results in Table 1.

As the results show, our model performs the best out of all the methods at analysing morphologically rich words with multiple morphemes. On these words, our model even outperforms Morfessor, which is explicitly designed as a morphological analyzer. This shows that our model learns splits which correspond well to human morphological analysis, even though we build no morphological knowledge into our model. However, when evaluating on all words, the Porter stemmer has a great advantage, as it is rule-based and able to give just the stem of words with great precision, which is effectively giving a canonical segmentation for words with just 2 morphemes.

We show some model analyses against the gold standard in Table 2.

4.2 Capturing semantic similarity

Next, we tested our model similarity scores against human similarity judgments. For these datasets, human annotators are asked to judge how similar two words are on a fixed scale. Model word vectors are evaluated based on ranking the word pairs according to their cosine similarity, and then measuring the correlation (using Spearman's ρ) between model judgments and human judgments (Levy et al., 2015).

Model	WordSim353	MEN Test	RW
PPMI-SVD	0.607	**0.601**	0.293
SGNS	**0.667**	0.557	**0.388**
C2V-NO-ATT	0.361	0.298	0.317
CHAR2VEC	0.345	0.322	0.282

Table 3: Similarity correlations of in-vocabulary word pairs between the models and human annotators.

Model	WordSim353	MEN Test	RW	RW OOV
C2V-NO-ATT	**0.358**	0.292	**0.273**	0.233
CHAR2VEC	0.340	**0.318**	0.264	**0.243**

Table 4: Similarity correlations of all word pairs between the character-level models and human annotators. RW OOV indicates results specifically on pairs in the RW dataset with at least one word not seen in the training corpus.

We use the WordSim353 dataset (Finkelstein et al., 2002), the test split of the MEN dataset (Bruni et al., 2014), and the Rare Word (RW) dataset (Luong et al., 2013). The word pairs in the WordSim353 and MEN datasets are typically simple, commonly occurring words denoting basic concepts, whereas the RW dataset contains many morphologically derived words which have low corpus frequencies. This is reflected by how many of the test pairs in each dataset contain out of vocabulary (OOV) items: 3/353 and 6/1000 of the word pairs in WordSim353 and MEN, compared with 1083/2034 for the RW dataset.

We report results for in-corpus word pairs in Table 3, and for all word pairs for those models able to predict vectors for unseen words in Table 4.

Overall, word-based embedding models learn vectors that correlate better with human judgments, particularly for morphologically simple words. However, character-based models are competitive with word-based models on the RW dataset. While the words in this dataset appear rarely in our corpus (of the in-corpus words, over half appear fewer than 100 times), each morpheme may be common, and the character-level models can use this information. We note that on the entire RW dataset (of which over half contain an OOV word), the character-based models still perform reasonably. We also note that on word pairs in the RW test containing at least one OOV word, the

	In-vocabulary			Out-of-Vocabulary	
	germany	*football*	*bible*	*foulness*	*definately*
Char2Vec unfiltered	*germaine*	*footballer*	*bibles*	*illness*	*definitely*
	germanies	*footballing*	*testament*	*seriousness*	*indefinitely*
	germain	*footballing*	*librarianship*	*sickness*	*enthusiastically*
	germano	*foosball*	*literature*	*loudness*	*emphatically*
	germaniae	*footballers*	*librarian*	*cuteness*	*consistently*
Char2Vec filtered	*poland*	*footballer*	*testament*	*illness*	*definitely*
	german	*basketball*	*literature*	*blindness*	*consistently*
	spain	*tennis*	*hebrew*	*consciousness*	*drastically*
	germans	*rugby*	*judaism*	*hardness*	*theoretically*
	france	*baseball*	*biblical*	*weakness*	*infinitely*

Table 5: Filtered and unfiltered model nearest neighbours for some in-vocabulary and out-of-vocabulary words

full Char2Vec model outperforms the C2V model without morphology. This suggests that character-based embedding models are learning to morphologically analyse complex word forms, even on unseen words, and that giving the model the capability to learn word segments independently helps this process.

We also present some word nearest neighbours for our Char2Vec model in Table 5, both on the whole vocabulary and then filtering the nearest neighbours to only include words which appear 100 times or more in our corpus. This corresponds to keeping the top 10k words, which is common among language models (Ling et al., 2015; Kim et al., 2016). We note that nearest neighbour predictions include words that are orthographically distant but semantically similar, showing that our model has the capability to learn to compose characters into word meanings.

We also note that word nearest neighbours seem to be more semantically coherent when rarely-observed words are filtered out of the vocabulary, and more based on orthographic overlap when the entire vocabulary is included. This suggests that for rarely-observed words, the model is basing its predictions on orthographic analysis, whereas for more commonly observed words it can ‘memorize’ the mapping between the orthography and word semantics.

4.3 Capturing syntactic and semantic regularity

Finally, we evaluate the structure of the embedding space of our various models. In particular, we test whether affixation corresponds to regular linear shifts in the embedding space.

Model	All Acc	Sem. Acc	Syn. Acc
PPMI-SVD	0.365	**0.444**	0.341
SGNS	**0.436**	0.339	0.513
C2V-NO-ATT	0.316	0.016	0.472
CHAR2VEC	0.355	0.025	**0.525**

Table 6: Results on the Google analogy task

To do this, we use the Google analogy dataset (Mikolov et al., 2013a). This consists of 19544 questions of the form “*A* is to *B* as *C* is to *X*”. We split this collection into semantic and syntactic sections, based on whether the analogies between the words are driven by morphological changes or deeper semantic shifts. Example semantic questions are on capital-country relationships (“*Paris* is to *France* as *Berlin* is to *X*”) and currency-country relationships (“*pound* is to *Great Britain* as *dollar* is to *X*”). Example syntactic questions are adjective-adverb relationships (“*amazing* is to *amazingly* as *apparent* is to *X*”) and opposites formed by prefixing a negation particle (“*acceptable* is to *unacceptable* as *aware* is to *X*”). This results in 5537 semantic analogies and 10411 syntactic analogies.

We use the method of Mikolov et al. (2013a) to answer these questions. We first ℓ_2-normalize all of our word vectors. Then, to answer a question of the form “*A* is to *B* as *C* is to *X*”, we find the word

w which satisfies

$$w = \underset{w \in V - \{a,b,c\}}{\operatorname{argmax}} \cos(w, b - a + c) \quad (4)$$

where $a,\ b,\ c$ are the word vectors for the words *A*, *B* and *C* respectively.

We report the results in Table 6. The most intriguing result is that character-level models are competitive with word-level models for syntactic analogy, with our Char2Vec model holding the best result for syntactic analogy answering. This suggests that incorporating morphological knowledge explicitly rather than latently helps the model learn morphological features. However, on the semantic analogies, the character-based models do much worse than the word-based models. This is perhaps unsurprising in light of the previous section, where we demonstrate that character-based models do worse at the semantic similarity task than word-level models.

5 Discussion

We only report results for English. However, English is a morphologically impoverished language, with little inflection and relatively few productive patterns of derivation. Our morphology test set reflects this, with over half the words consisting of a simple morpheme, and over 90% having at most 2 morphemes.

This is unfortunate for our model, as it performs better on words with richer morphology. It gives consistently more accurate morphological analyses for these words compared to standard baselines, and matches word-level models for semantic similarity on rare words with rich morphology. In addition, it seems to learn morphosyntactic features to help solve the syntactic analogy task. Most of all, it is language-agnostic, and easy to port across different languages. We thus expect our model to perform even better for languages with a richer morphology than English, such as Turkish and German.

6 Conclusion

In this paper, we present a model which learns morphology and word embeddings jointly. Given a word, it splits the word in to segments and ranks the segments based on their context-predictive power. Our model can segment words into morphemes, and also embed the word into a representation space.

We show that our model is competitive at the task of morpheme boundary recovery compared to a dedicated morphological analyzer, beating dedicated analyzers on words with a rich morphology. We also show that in the representation space word affixation corresponds to linear shifts, demonstrating that our model can learn morphological features.

Finally, we show that character-level models, while outperformed by word-level models generally at the task of semantic similarity, are competitive at representing rare morphologically rich words. In addition, the character-level models can predict good quality representations for unseen words, with the morphologically aware character-level model doing slightly better.

References

Harald R. Baayen, Richard Piepenbrock, and Leon Gulikers. 1995. *The CELEX Lexical Database. Release 2 (CD-ROM).* Linguistic Data Consortium, University of Pennsylvania, Philadelphia, Pennsylvania.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. *EMNLP*.

David A. Balota, Melvin J. Yap, Keith A. Hutchison, Michael J. Cortese, Brett Kessler, Bjorn Loftis, James H. Neely, Douglas L. Nelson, Greg B. Simpson, and Rebecca Treiman. 2007. The english lexicon project. *Behavior Research Methods*.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *ICML*, Beijing, China, jun. *Award for best application paper*.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.*

John A. Bullinaria and Joseph P. Levy. 2007. Extracting semantic representations from word cooccurrence statistics: A computational study. *Behavior Research Methods*.

Franois Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

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

Ryan Cotterell, Tim Vieira, and Hinrich Schütze. 2016. A joint model of orthography and word segmentation. In *NAACL*.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3:1–3:34, February.

Ferdinand de Saussure. 1916. *Course in General Linguistics*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, June.

Sharon Goldwater, Mark Johnson, and Thomas L. Griffiths. 2005. Interpolating between types and tokens by estimating power-law generators. In Y. Weiss, B. Schlkopf, and J. Platt, editors, *NIPS*, pages 459–466. MIT Press, Cambridge, MA.

Zelig S. Harris. 1954. Distributional structure. *Word*.

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. *AAAI*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR*.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. *ACL*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *EMNLP*.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *ICLR Workshop*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *NIPS*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273. Curran Associates, Inc.

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

Radu Soricut and Franz Ochs. 2015. Unsupervised morphology induction using word embeddings. *NAACL*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *In Proceedings of NAACL*.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. CCG supertagging with a recurrent neural network. *ACL*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *NIPS*.

George Kingsley Zipf. 1935. *The psycho-biology of language*.