Instructions for EMNLP 2020 Proceedings

Anonymous EMNLP submission

Abstract

The analogy task introduced by Mikolov et al. (2013) has become the standard method for tuning the hyperparameters for word embedding models. It has an intrinsic appeal and has been shown to correlate well with downstream tasks. In this paper, however, we show that the analogy task is unsuitable for low-resource settings for two reasons: (1) it requires large amounts of data to make measurable progress on, and (2) it is not even well-defined in some settings. We solve this problem by introducing the `OddOneOut` and `Topk` tasks, which are specifically designed for model selection in the low-resource setting. Performance on these tasks also correlates well with downstream tasks, but these tasks are more sensitive to word embedding performance when trained on smaller datasets. We use these metrics to successfully tune hyperparameters for 18 low-resource languages provided by the Classical Languages ToolKit. The smallest of these languages (Ancient Gujarati) has only 1813 unique tokens available.

1 Introduction

There has been a strong push in recent years for improved natural language processing on low-resource languages. Little data exists for these languages, and so trained machine learning models have low accuracies. Recent work has focused on solving this problem by developing more sample-efficient algorithms that can achieve higher accuracies with less data. We argue, however, that standard evaluation tasks are not sensitive enough to measure model quality in the low-resource setting, and new performance measures tailored to the low-resource setting are required. Specifically, we argue that the standard analogy task is not suitable for the low-resource setting and we introduce two new evaluation metrics designed for the low-resource setting.

Mikolov et al. (2013) introduced the word2vec model for training word embeddings and the analogy task for evaluating them. In the analogy task, the goal is to answer questions similar to

```
Man is to King as Woman is to ?
```

by exploiting the linear structure of the word embeddings. Follow on work has focused on developing better algorithms for training word embeddings as measured by performance on the analogy task. GloVe (Pennington et al., 2014) and FastText (Bojanowski et al., 2016) are the two most prominent examples, although dozens of alternatives have now been proposed. The analogy task has proven so successful for English language research because it correlates well with performance on a wide range of downstream tasks (Wang et al., 2019).

While most word embeddings algorithms focus on the high-resource domain, several recent works have developed new algorithms designed for training models in the low-resource setting (Adams et al., 2017; Jiang et al., 2018; Gupta et al., 2019; Jungmaier et al., 2020). Unfortunately, each of these works evaluate their method only in a simulated low-resource environment using English text and not on any actual low-resource languages. They do this specifically because no evaluation metrics exist that are suitable for their low-resource target languages. They also do not use the analogy task to measure performance. Instead, they use word similarity tasks. These tasks include datasets based on the English-language WordNet: WordSim Similarity (Zesch et al., 2008)WordSim Relatedness (Agirre et al., 2009), and datasets based on human annotation of English language word pairs: WordSim363 (Finkelstein et al., 2001), Mechanical Turk (Radinsky et al., 2011), MEN (Bruni et al., 2012). These word similarity tasks are more sensitive to their low-resource experimental design than the analogy task, but they cannot be easily used

in non-English languages because they require the manual generation of new datasets for each language.

Perhaps surprisingly, there is growing body of digital humanities work where English language text is low-resource. This occurs when researchers attempt to compare how to bodies of text use words differently over different time periods (e.g. Hamilton et al., 2016b,a; Kutuzov et al., 2018; Kozlowski et al., 2019; Dubossarsky et al., 2017; Tang, 2018; Szymanski, 2017; Liang et al., 2018; Chen et al., 2017) or different political ideologies Azarbonyad et al. (2017) or different twitter users (Kulkarni et al., 2015).

Our contributions can be summarized with the following three points.

1. We introduce the first word embedding evaluation tasks designed specifically for the low-resource setting, `OddOneOut` and `Topk`. Code for computing these metrics is released as an open source Python library.[1]

2. We introduce a method for automatically generating test datasets for the `OddOneOut` and `Topk` tasks in the 581 languages supported by the wikidata project.[2] All previous evaluation methods work on only a small number of languages (typically just English), and require significant manual work to adapt to new languages.

3. We perform the largest existing multilingual evaluation on low-resource languages. Specifically, we provide word embeddings for the 18 languages of the Classical Languages ToolKit (CLTK) library ***FIXME:*** *bibtex format is wrong (et al., 2014–2019)*. We further introduce a new downstream task called the Language Comparison Task (LCT) that lets us map which topics are included in these classical language corpora.

The remainder of the paper is organized as follows. Section 2 formally defines the `Topk` and `OddOneOut` tasks. Section 3 empirically demonstrates that these tasks are better than the analogy task in low-resource settings. We use a synthetic English language experimental design common

[1] URL hidden for blind review.

[2] For the full list of languages supported, see `https://www.wikidata.org/wiki/Help:Wikimedia_language_codes/lists/all` ***FIXME:*** *You wrote 461 without a citation. Which is correct?*

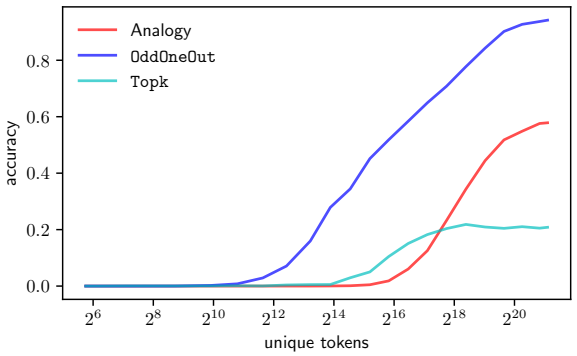

Figure 1: The plot above compares the regions of effectiveness for our evaluation metrics. The analogy task fails to measure change in accuracy of the embeddings until the number of unique words in the training dataset reaches 2^{16}, much later than both `OddOneOut` and `Topk`. Though `OddOneOut` seems the clear victor of these methods, experimentation in 3 shows that `Topk` works better in some circumstances.

in previous work, and demonstrate the versatility of these evaluation metrics by applying them to an emoji embedding task for which the analogy task is not even well defined. Section 4 computes word embeddings for the 18 languages provided by CLTK. We further introduce our technique for using wikidata to automatically generate the test sets for the `OddOneOut` and `Topk` tasks and the LCT task and provide a semantic analysis of the topics covered in each of these 18 language copora. Section 5 concludes by discussing how extensions to this work could serve communities working with low-resource languages.

2 Evaluation Methods

The `OddOneOut` and `Topk` tasks are simple and generally applicable. Both tasks require a test set in the same easy to generate format. The test set consists of a list of categories, and each category contains a list of words that belong to that category. For example, the category `fruit` might contain the words `banana`, `apple`, and `orange`. ***FIXME:*** *use a real example.* The `OddOneOut` task measures a model's ability to identify words that are unrelated to the category, and the `Topk` task measures a model's ability to identify words that are related to the category.

Formally, assume that there are m categories, that each category has n words[3], that there are v

[3] In practice, the number of words per category can vary for each category, however for notational simplicity we assume that each category has the same number of words.

total words in the vocabulary, and that the words are embedded into $\mathbb{R}^d$. Let $c_{i,j}$ be the jth word in category i, and let $C_i = \{c_{i,1}, c_{i,2}, ..., c_{i,n}\}$ be the ith category.

2.1 The `Topk` method

Let $\texttt{Sim}(k, w)$ return the k most similar words in the vocabulary to w. We use the cosine distance in all our experiments, but any distance metric can be used. Next, define the `Topk` score for class i to be

$$\texttt{Topk}(k, i) = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{k}\sum_{x\in\texttt{Sim}(k,c_{i,j})}\mathbb{1}[x \in C_i] \quad (1)$$

and the `Topk` score for the entire evaluation set to be

$$\texttt{Topk}(k) = \frac{1}{m}\sum_{i=1}^{m}\texttt{Topk}(k, i). \quad (2)$$

The runtime of `Sim` is $O(dvk)$.[4] So the runtime of $\texttt{Topk}(k, i)$ is $O(dnk^2v)$ and the runtime of $\texttt{Topk}(k)$ is $O(dnmk^2v)$. Typically k is small (we recommend $k = 3$ in our experiments), and so the runtime is linear in all of the interesting parameters. In particular, it is linear in both the size of our vocabulary, the number of categories in the test set, and the size of the categories.

2.2 The `OddOneOut` method

Define the `OddOneOut` score of a set S with k words with respect to a word $w \notin S$ as

$$\texttt{OddOneOut}(S, w) = \mathbb{1}[w = \hat{w}], \quad (3)$$

where

$$\hat{w} = \arg\max_{x\in S\cup\{w\}}\|x - \mu\| \quad (4)$$

and

$$\mu = \frac{1}{k+1}\left(w + \sum_{i=1}^{k} s_i\right). \quad (5)$$

We define the kth order `OddOneOut` score of a category i to be

$$\texttt{OddOneOut}(k, i) = \frac{1}{|P|}\sum_{(S,w)\in P}\texttt{OddOneOut}(S, w) \quad (6)$$

where

$$P = \{(S, w) : S \text{ is a combination of } k \text{ words from } C_i, \text{and } w \in V - C_i\}. \quad (7)$$

In Equation (7) above, the total number of values that S can take is $\binom{n}{k} = O(n^k)$, and the total number of values that w can take is $O(v)$, so $|P| = O(n^k v)$. Finally, we define the k-th order `OddOneOut` score of the entire evaluation set to be

$$\texttt{OddOneOut}(k) = \frac{1}{m}\sum_{i=1}^{m}\texttt{OddOneOut}(k, i). \quad (8)$$

The runtime of $\texttt{OddOneOut}(S, w)$ is $O(dk)$. So the runtime of $\texttt{OddOneOut}(k, i)$ is $O(dkn^k v)$ and the runtime of $\texttt{OddOneOut}(k)$ is $O(dkmn^k v)$.

Comparing the runtimes of $\texttt{OddOneOut}(k)$ and $\texttt{Topk}(k)$, we can see that $\texttt{OddOneOut}(k)$ is a factor of $O(m^{k-1}k^{-1})$ slower. In real world applications, $m \gg k$, and so `OddOneOut` will take considerably more time to compute. In particular, the Mikolov test evaluations in Section 3 below use $m = 50$ and $k = 3$, so the $\texttt{OddOneOut}(k)$ score takes approximately 800x longer to compute.

To solve this problem, we use a sampling strategy. Let $\tilde{P}$ denote the set of p samples without replacement from the set P. Then we rewrite Equation 6 as

$$\texttt{OddOneOut}(k, i) = \frac{1}{p}\sum_{(S,w)\in\tilde{P}}\texttt{OddOneOut}(S, w) \quad (9)$$

With this definition, the runtime of $\texttt{OddOneOut}(k)$ is $O(dkmpv)$, which is linear in all the parameters of interest. ***FIXME:*** *In our experiments, we found $p = 1000$ to give sufficiently accurate results without taking too much computation.*

3 Experiments

We demonstrate the usefulness of our evaluation metrics with two experiments. First, we show that the `OddOneOut` and `Topk` metrics are better measures of word embedding quality than the `Analogy` metric in the low-resource regime. Second, we show that the `OddOneOut` and `Topk` metrics are useful for model selection in an emoji embedding task where the analogy task cannot be applied. This experiment also demonstrates that

[4] We use gensim's implementation of `Sim`, which uses the naive loop strategy for computing the nearest neighbor. Data structures like the kd-tree or cover tree could potentially be used to speed up this search, but we did not find such data structures necessary.

the OddOneOut and Topk metrics correlate with downstream task performance.

3.1 English Experiments

This experiment measures the performance of the OddOneOut, Topk, and Analogy metrics as a function of data set size.

For training data, we use a 2017 dump of the English-language Wikipedia that contains 2 billion total tokens and 2 million unique tokens. The dataset is freely distributed with the popular gensim library (Řehůřek and Sojka, 2010) for training word embeddings, and it is therefore widely used. State-of-the-art embeddings are trained on significantly larger datasets—for example, datasets based on the common crawl contain hundreds of billions of tokens even for non-English languages (Buck et al., 2014; Grave et al., 2018)—but since our emphasis is on the low-resource setting, this 2 billion token dataset is sufficient.

Using the wikipedia dataset, we generate a series of synthetic low-resource datasets of varying size. First, we sort the articles in the wikipedia dataset randomly.[5] Then, each dataset i contains the first 2^i tokens in the randomly ordered wikipedia dump.

On each of these low-resource datasets, we train a word2vec skipgram model with gensim's default hyperparameters[6], which are known to work well in many contexts. Importantly, we do not tune these hyperparameters for each low-resource dataset. Instead, we use the same hyperparameters because our goal is to isolate the effects of dataset size on the three evaluation metrics.

For the analogy task, we use the standard Google Analogy Test Set introduced by (Mikolov et al., 2013). This test set contains 14 sets of analogies, and each analogy set contains 2 categories that are being compared. For example, the countries-capitals analogy set which has analogies like

$$\texttt{England} - \texttt{London} + \texttt{Paris} \approx \texttt{France} \quad (10)$$

contains both a list of countries and a list of capitals. We generate test sets for the OddOneOut and Topk tasks from the 28 categories in the Google test set.

[5] This sorting is required for our low-resource datasets to be representative of English language text. Without this random sorting step, most of our datasets would be based only on articles that begin with the letter A, and therefore would not contain a representative sample of English words.

[6] Embedding dimension 100, number of epochs 1, learning rate 0.025, window size is 5, min count is 5. ***FIXME:*** *should we defined min count?*

The results are shown in Figure 1. ***FIXME:*** *The OddOneOut and Topk metrics show higher results*

3.2 Emoji Experiments

This second experiment demonstrates the versatility of our methods by applying them to the domain of emoji embeddings. We show that our generic Topk and OddOneOut metrics perform as well as a custom designed emoji evaluation metric, but our metrics are more versatile.

Emoji embeddings are an important topic of study because they are used to improve the performance of sentiment analysis systems (e.g. Eisner et al., 2016; Felbo et al., 2017; Barbieri et al., 2017; Ai et al., 2017; Wijeratne et al., 2017; Al-Halah et al., 2019). Unfortunately, the standard analogy task is not suitable for evaluating the quality of emoji embeddings for two reasons. First, emoji embeddings are inherently low-resource—only 3000 unique emojis exist in the Unicode standard—and thus evaluation techniques specifically designed for the low-resource setting will be more effective. Second, the semantics of most emojis do not allow them to be used in any analogy task. In particular, the original emoji2vec paper (Eisner et al., 2016) identifies only ***FIXME:*** *5?* possible emoji analogies.

In order to tune their emoji embeddings, Eisner et al. (2016) therefore do not use the analogy task, and instead introduce an ad-hoc "emoji-description classification" metric that required the creation of a test set with manually labeled emotion-description pairs. Due to the expense of manually creating this test set, only ***FIXME:*** *64?* of the 3000 Unicode emojis are included. The Topk and OddOneOut metrics improve on the "emoji-description classification" metric because they are able to evaluate the quality of all emojis and require no manual test set creation. For our test set categories, we use the categories that the Unicode standard provides for each emoji.[7]

To test the performance of the three metrics, we use them to tune the hyperparameters of an emoji2vec model. To ensure the fairest comparison possible, we use the original emoji2vec code for training and model selection, changing only the function call to the metric used. In particular, this means we are only embedding and evaluating on

[7] For the full list of categories, see https://unicode.org/Public/emoji/13.0/emoji-test.txt

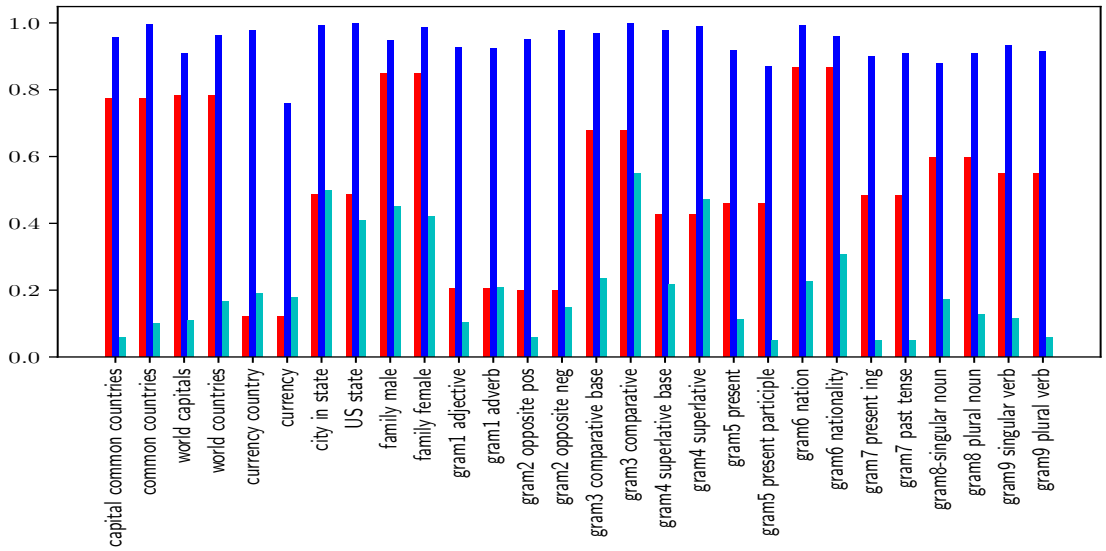

Figure 2: The methods perform better on some categories than others. `Topk` seems to excel in categories that are more homogenous like 'family female', while analogies seem to work best with geographical relationships. Note that in adapting the Google analogy set to work with our methods required splitting each relationship pair into two separate categories. As a result the analogy score for a given relationship is shown twice; one bar in each of the categories that make up the pair.

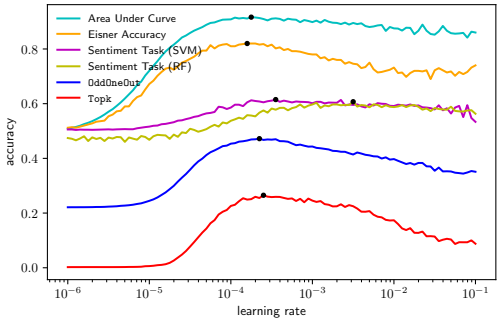

Figure 3: This figures shows the tuning of emoji embeddings across different learning learning rates where the max accuracy for each metric is marked with a point. Both `Topk` and `OddOneOut` follow the shape of (Eisner et al., 2016) area under the curve and accuracy metrics. Our methods lead us to select essentially the same hyperparameters as (Eisner et al., 2016) and reproduced results on the downstream task.

the subset of ***FIXME:*** *64?* emojis supported by the "emoji-description classification" metric. The code allows tuning the model's learning rate, ***FIXME:*** *insert other options here*. We found that the learning rate was the only hyperparameter to have a significant impact, and Figure 3 shows how the learning rate effects the performance on the three evaluation metrics and the performance on a downstream sentiment analysis task. All three metrics show optimal performance with a learning rate of approximately 8×10^{-4}, which also results in the best performance on the downstream task. This indicates that our `Topk` and `OddOneOut` metrics generate the same models as the specialized "emoji-description classification" metric, but our metrics have the advantage of being simpler, more widely applicable, and easier to generate test data for.

FIXME: *Explain model selection and that comparison of metrics should not be done.*

4 Multilingual Content Analysis

In this section we perform the first highly multilingual analysis of word embeddings for low-resource languages. We analyze 18 languages provided by the Classical Languages ToolKit (CLTK) library (et al., 2014–2019).[8] Each of these languages is

[8] Specifically, we analyzed all languages for which CLTK provides both a training corpus and a tokenization function, as these are the minimum requirements needed for training word

"extinct" in the sense that no new native text will ever be generated in these languages. To get better models on these languages, it is impossible to collect more data, we must develop better techniques for the low-resource setting. The largest of these language datasets is Ancient Greek, with 37.8 million tokens, and the smallest is Ancient Gujarati with only 1813 tokens.

First we describe a procedure for automatically generating test set data for the `OddOneOut` and `Topk` tasks using wikidata. Then, we describe our model training and selection procedure for each language. Finally, we perform the LCT task and perform an interlanguage analysis of the corpora's content.

4.1 Test Set Generation with Wikidata

One of the most difficult and time consuming steps in the process of generating of high quality word embeddings is the creation of a comprehensive test set. Consequently, one of the biggest advantages of the `OddOneOut` and `Topk` methods is that compatible test sets can be semi-automatically generated in hundreds of languages. This is possible using Wikidata. This publicly available SPARQL database contains a comprehensive structure of the semantic content contained in Wikipedia along with its relationship to other items. Simple queries constructed using the Wikidata Query Service can be used to return semantic categories of words that can be included in a test set.

Using Wikidata we were able to reconstruct all of the semantic categories contained within the Google analogy set with the option of greater customization and more categories. Additionally, Wikidata supports the translation of queried items into 461 languages, allowing test sets to easily be converted between languages. Though not all items have translated labels in all 461 languages, support is quite extensive and will only get better. ***FIXME:*** *Is 461 correct?*

One of the disadvantages of using Wikidata is that syntactic categories are much harder to construct; however, since semantic categories are usually more difficult to generate and require more arbitrary decisions this approach is still very valuable.

FIXME: *What are the exact categories used?*

FIXME: *Does the library allow the easy construction of these categories?*

embeddings.

4.2 Model Selection

There are 7 hyperparameters for our language models, and we use the random search method (Bergstra and Bengio, 2012) to tune these hyperparameters. Random search is simple to implement, computationally efficient, easy to parallelize, and avoids the curse of dimensionality inherent to grid search and Bayesian optimization methods. ***FIXME:*** *Table 2 defines the hyperparameters and the range of values sampled for each. Should this be a table or included directly in the text?*

To ensure a fair comparison, we randomly sampled 100 sets of hyperparameter combinations.[9] Then train each language on this same set of hyperparameters. The best results are reported in Table 1. The optimal set of hyperparameters is different for each language, which underscores the importance of proper model tuning in the low-resource regime. Whereas all previous highly multilingual work used the same set of hyperparemeters for all models, we show that this is not an optimal strategy.

4.3 Language Comparison Task

We now introduce a novel downstream task for word embeddings called the *Language Comparison Task* (LCT). The goal of this task is to better visualize the topics written about in our 18 language corpora. We choose three religious topics: Biblical Figures, Facets of Buddhism, and Facets of Hinduism. Each of these topics has an entry in Wikidata, ***FIXME:*** *I'm not sure what the correct wikidata terminology is here.* and so it is easy to create categories based on these topics in each language. Figure 4 provides a list of words associated with each topic.

For each language and topic, we compute the `Topk` and `OddOneOut` metrics and plot the results in Figure 4. ***FIXME:*** *explain significance*

Perhaps more interesting, this downstream task also gives rise to circumstances in which `Topk` performance exceeded `OddOneOut`. In some cases (like Middle High German on Biblical Figures) `Topk` merely surpasses `OddOneOut` and in others (such as Gujarati on Facets of Hinduism) it is the only metric to achieve a performance score. It is likely that a complex interaction between the model and the category being evaluated on can result in either method achieving better performance, thus it is valuable to have both methods at our dis-

[9] We found that 100 combinations was sufficient to give good results without being too computationally burdensome.

	Corpus				Parameters							Scores	
	Language	Test Set	Tokens	Unique Tokens	Model	Type	Dim	Window	LR	Min Count	Lemma	OddOneOut	Topk	Comb
Hellenic	greek	ancient greek	37 868 209	1 877 574	w2v	cbow	40	9	-1	4	False	1140	8	
	greek	modern greek	37 868 209	1 877 574	fast	sg	90	7	-1	5	True	300	9	
Italic	latin	latin	17 777 429	470 790	w2v	cbow	50	10	-1	7	False	2748	48	
	old french	french	68 741	8343	fast	sg	250	6	-1	8	False	1	7	
Germanic	middle english	english (old)	7 048 144	314 527	fast	sg	90	5	-1	7	False	239	7	
	middle high german	german	2 090 954	60 674	fast	cbow	15	6	-1	3	False	9	19	
	old english	english (old)	104 011	33 018	fast	cbow	425	3	-1	3	True	0	1	
	old norse	icelandic	458 377	59 186	w2v	cbow	60	10	-1	3	False	968	2	
	old swedish	swedish	1 297 740	116 374	fast	sg	50	8	-1	5	False	50	1	
Indo-Aryan	bengali	bengali	5539	2323	fast	cbow	15	3	-1	4	False	0	2	
	gujarati	gujarati	1813	1140	fast	sg	80	3	-1	5	False	0	7	
	hindi	hindi	587 655	55 483	fast	cbow	45	4	-1	8	False	263	2	
	malayalam	malayalam	9235	5405	-	-	-	-	-	-	-	-	-	
	marathi	marathi	797 926	96 778	w2v	sg	400	4	-1	6	False	342	1	
	punjabi	punjabi	1 024 075	31 343	fast	sg	50	8	-1	5	False	0	1	
	sanskrit	sanskrit	4 042 204	896 480	w2v	sg	35	9	-1	10	False	1530	1	
	telugu	telugu	537 673	276 330	w2v	cbow	60	10	-1	3	False	50	0	
Semitic	classical arabic	arabic	81 306	20 493	-	-	-	-	-	-	-	-	-	
	hebrew	hebrew	41 378 460	893 512	fast	sg	30	4	-1	3	False	1098	6	

Table 1: The table above provides details for the best model trained for languages supported by CLTK. Following a tuning process, models were chosen by their `Combined Score` which is calculated as the harmonic mean of `Topk` and `OddOneOut`. It is important to note that the absolute score for our evaluation metrics are not important in and of themselves, rather they are important as indicators of a change in embedding quality that the analogy task would fail to show. To emphasize this, we have reported the raw number of correct answers for each metric. Using `OddOneOut` and `Topk` allows us to tune models trained on corpora with unique token counts in the thousands instead of the millions.

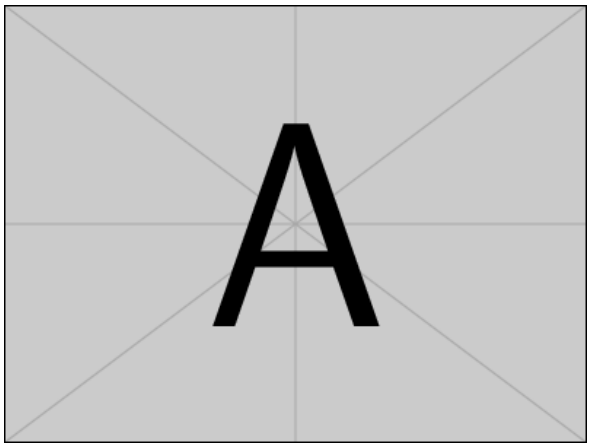

Table 2: Hyperparameters.

posal. Since this exact relationship is still not fully understood we tune and evaluate models using both metrics and compute the harmonic mean to choose the most accurate model.

FIXME: *There's a simpler way to measure how much each corpus discusses each category: simply count the number of times each word from the category is used. Can we explain why our method is better than this simpler method?*

5 Discussion

FIXME: *Don't actually make any changes here. The notes in this section are just for me.*

FIXME: *Michel et al. (2020) study Hiligaynon*

Not yet using wikidata to its fullest extent. Wikidata categories form a hierarchy. More accurate evaluations could be done by using all categories in the hierarchy, weighting top-level categories higher than leaf categories. This has computational challenges and so we reserve this for future work.

FIXME: *Where should this go? Gonen et al. (2020) propose a new method for comparing the similarity of two word embeddings in order to find words that are used differently. Their method assumes that quality word embeddings have already been trained, and our evaluation metrics provide a way to ensure that this training is done well.*

FIXME: *Where should this go? Camacho-Collados and Navigli (2016) propose the* `OutlierDetection` *metric that is superficially similar to our* `OddOneOut` *metric. Differences include: their metric is designed to evaluate only a single category relative to itself and is therefore relatively efficient to compute. Our metric is designed to evaluate many categories relative to all words in the training data. A naive computation is therefore much more expensive to compute, and so we introduce a sampling strategy to make the metric computationally feasible. We also demonstrate that our metric is suitable for the low-resource regime (they only evaluate on datasets with billions of tokens) and make our method available in an easy-to-use open source*

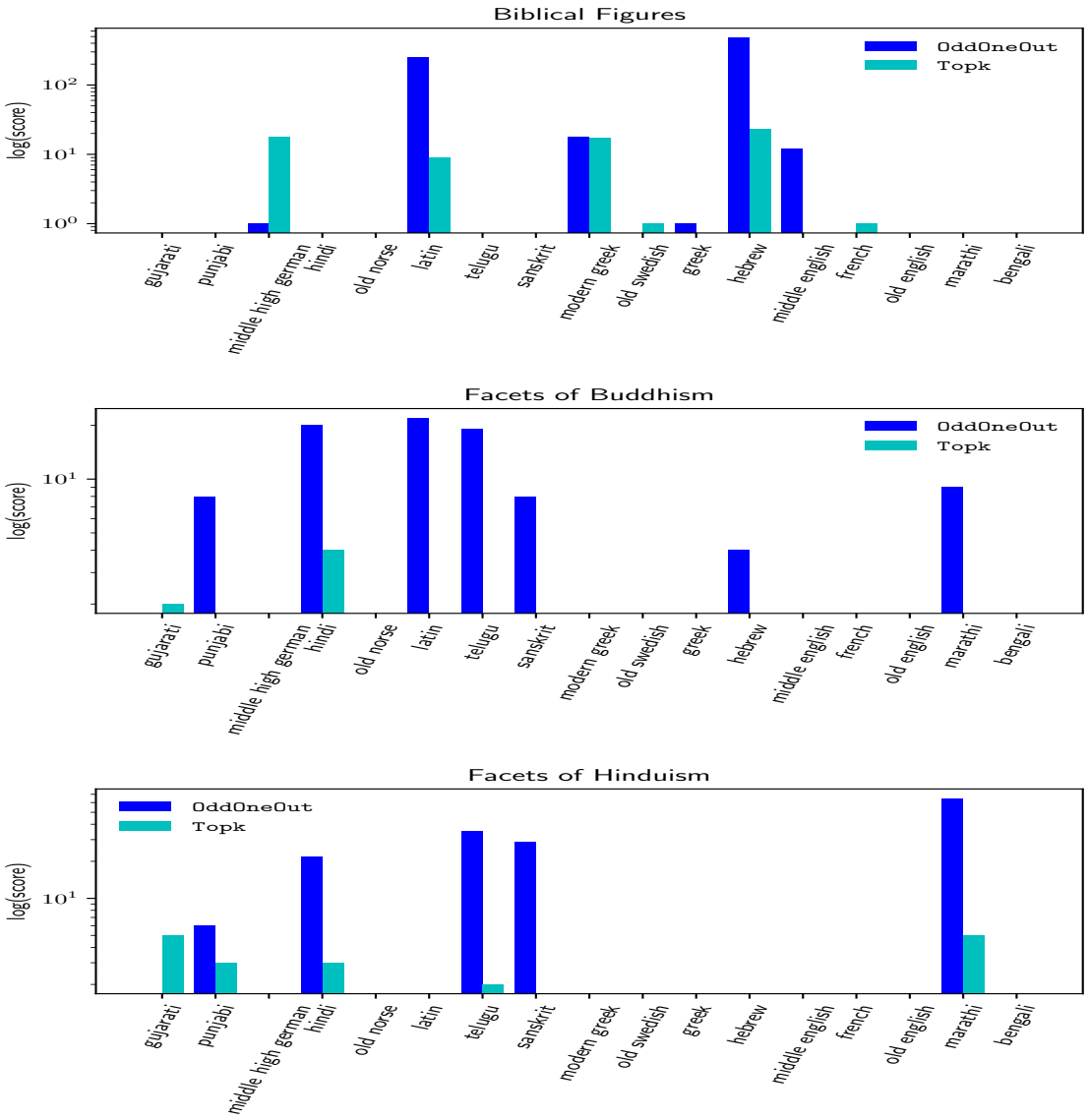

Figure 4: (**Top**) High performance on the Biblical Figures category indicates some level of biblical influence via the corpus. Interestingly, we see that greek embeddings optimized on the Modern Greek test set significantly outperformed the embeddings optimized for the Ancient Greek. This matches our intuition that things of a biblical nature have had a greater influence on Modern Greek than Ancient Greek. (**Middle**) Though many of the Indic language embeddings performed well on the Facets of Buddhism, surprisingly so did our Latin and Hebrew embeddings. This leads us to believe that some Buddhist concepts and words are shared by corpora spanning languages as diverse as Hebrew, Latin, and Hindi. At the same time, it should also be noted that Latin and Hebrew were two of the largest models trained compared to other classical languages and thus also likely benefit from greater resource richness. (**Bottom**) Similar to Figure **??** we see languages more closely related to the topic of the category achieving better performance, in this case primarily the Indic languages. Interstingly, our Bengali embeddings performed poorly on this category suggesting that the corpus did not refer heavily to the Hindu context.

python library.

FIXME: *Add this? Recent work Guntuku et al. (2019) combines uses the subdivision technique (Section **??**) to study how emoji are used differently in different parts of the world. This technique turns high-resource tasks into low-resource tasks, and so if you start with a low-resource task it will become even more low-resource.*

FIXME: *Where does this go?*

Al-Rfou et al. (2013) trained word2vec embeddings on 100 different languages using Wikipedia as the training data. Grave et al. (2018) extended this work by training FastText embeddings on 157 languages using data from the Common Crawl project.

1. *For both papers, the Hindi language had the smallest amount of training data, with 23 million and 1.8 billion tokens, respectively. In Section **??**, we consider languages with significantly less training data. For example, the Ancient Hindi language contains only 0.6 million tokens, and we are still able to generate meaningful word embeddings using our evaluation metrics.*

2. *Due to the difficulty of evaluating so many languages, however, the authors evaluate the embeddings only on 10 languages, and the other 147 remain unevaluated.*

FIXME: *where does this go? Bias-variance tradeoff in dimensionality of word embeddings (Yin and Shen, 2018)*

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 937–947, Valencia, Spain. Association for Computational Linguistics.

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches.

Wei Ai, Xuan Lu, Xuanzhe Liu, Ning Wang, Gang Huang, and Qiaozhu Mei. 2017. Untangling emoji popularity through semantic embeddings. In *Eleventh International AAAI Conference on Web and Social Media*.

Kyle P. Johnson et al. 2014–2019. Cltk: The classical language toolkit. https://github.com/cltk/cltk. DOI 10.5281/zenodo.593336.

Ziad Al-Halah, Andrew Aitken, Wenzhe Shi, and Jose Caballero. 2019. Smile, be happy:) emoji embedding for visual sentiment analysis. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. *arXiv preprint arXiv:1307.1662*.

Hosein Azarbonyad, Mostafa Dehghani, Kaspar Beelen, Alexandra Arkut, Maarten Marx, and Jaap Kamps. 2017. Words are malleable: Computing semantic shifts in political and media discourse. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1509–1518.

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? *arXiv preprint arXiv:1702.07285*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145.

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. 2014. N-gram counts and language models from the common crawl. In *LREC*, volume 2, page 4. Citeseer.

José Camacho-Collados and Roberto Navigli. 2016. Find the word that does not belong: A framework for an intrinsic evaluation of word vector representations. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 43–50, Berlin, Germany. Association for Computational Linguistics.

Baitong Chen, Satoshi Tsutsui, Ying Ding, and Feicheng Ma. 2017. Understanding the topic evolution in a scientific domain: An exploratory study for the field of information retrieval. *Journal of Informetrics*, 11(4):1175–1189.

Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. 2017. Outta control: Laws of semantic change and inherent biases in word representation models. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 1136–1145.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

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*, pages 406–414.

Hila Gonen, Ganesh Jawahar, Djamé Seddah, and Yoav Goldberg. 2020. Simple, interpretable and stable method for detecting words with usage change across corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 538–555.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. *arXiv preprint arXiv:1802.06893*.

Sharath Chandra Guntuku, Mingyang Li, Louis Tay, and Lyle H Ungar. 2019. Studying cultural differences in emoji usage across the east and the west. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 13, pages 226–235.

Vishwani Gupta, Sven Giesselbach, Stefan Rüping, and Christian Bauckhage. 2019. Improving word embeddings using kernel PCA. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 200–208, Florence, Italy. Association for Computational Linguistics.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016a. Cultural shift or linguistic drift? comparing two computational measures of semantic change. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 2116. NIH Public Access.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016b. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096*.

Chao Jiang, Hsiang-Fu Yu, Cho-Jui Hsieh, and Kai-Wei Chang. 2018. Learningword embeddings for low-resource languages by pu learning. *arXiv preprint arXiv:1805.03366*.

Jakob Jungmaier, Nora Kassner, and Benjamin Roth. 2020. Dirichlet-smoothed word embeddings for low-resource settings. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3560–3565, Marseille, France. European Language Resources Association.

Austin C Kozlowski, Matt Taddy, and James A Evans. 2019. The geometry of culture: Analyzing the meanings of class through word embeddings. *American Sociological Review*, 84(5):905–949.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635.

Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. 2018. Diachronic word embeddings and semantic shifts: a survey. *arXiv preprint arXiv:1806.03537*.

Shangsong Liang, Xiangliang Zhang, Zhaochun Ren, and Evangelos Kanoulas. 2018. Dynamic embeddings for user profiling in twitter. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1764–1773.

Leah Michel, Viktor Hangya, and Alexander Fraser. 2020. Exploring bilingual word embeddings for Hiligaynon, a low-resource language. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2573–2580, Marseille, France. European Language Resources Association.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Terrence Szymanski. 2017. Temporal word analogies: Identifying lexical replacement with diachronic word embeddings. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 2: short papers)*, pages 448–453.

Xuri Tang. 2018. A state-of-the-art of semantic change computation. *arXiv preprint arXiv:1801.09872*.

Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C-C Jay Kuo. 2019. Evaluating word embedding models: Methods and experimental results. *APSIPA transactions on signal and information processing*, 8.

Sanjaya Wijeratne, Lakshika Balasuriya, Amit Sheth, and Derek Doran. 2017. A semantics-based measure of emoji similarity. In *Proceedings of the International Conference on Web Intelligence*, pages 646–653.

Zi Yin and Yuanyuan Shen. 2018. On the dimensionality of word embedding. In *Advances in Neural Information Processing Systems*, pages 887–898.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using wiktionary for computing semantic relatedness.