Evaluating Word Embeddings on Low-Resource Languages

Anonymous EMNLP submission

Abstract

The analogy task introduced by Mikolov et al. (2013) has become the standard metric for tuning the hyperparameters of word embedding models. In this paper, however, we argue that the analogy task is unsuitable for low-resource languages for two reasons: (1) it requires that word embeddings be trained on large amounts of text, and (2) analogies may not be well-defined in some low-resource settings. We solve these problems by introducing the `OddOneOut` and `Topk` tasks, which are specifically designed for model selection in the low-resource setting. We use these metrics to successfully tune hyperparameters for a low-resource emoji embedding task and word embeddings on 16 extinct languages. The largest of these languages (Ancient Hebrew) has a 41 million token dataset, and the smallest (Ancient Gujarati) has only a 1813 token dataset.

1 Introduction

Imagine you're given the task of training a text classification model for Middle English. This form of English was spoken in the Middle Ages from 1066-1500 CE. It is significantly different from modern English (Chamonikolasová, 2014), and only a handful of historians speak this language today.

A natural first step would be to train word embeddings. So you use the Classical Languages ToolKit (CLTK) (Johnson, 2014) to download the largest corpus of known Middle English documents (only 7 million tokens, 0.3 million unique tokens), and GenSim (Řehůřek and Sojka, 2010) to train the embeddings. To evaluate the embeddings, you follow the current standard practice established by Mikolov et al. (2013) of using an analogy test set. Of course, you can't use Mikolov et al. (2013)'s test set—it is in English, and Middle English is not English. But you also can't even use translations of their test set—many of the analogy concepts simply didn't exist in the Middle Ages. For example, the analogy

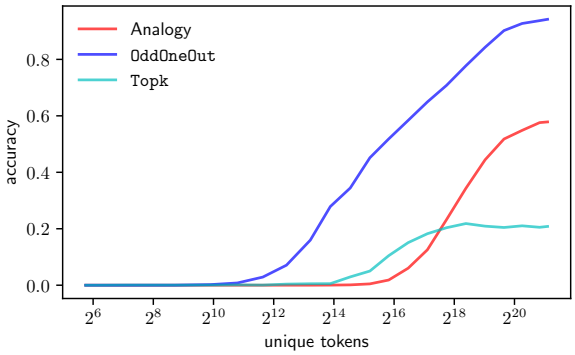

Figure 1: The standard analogy task (Mikolov et al., 2013) fails to measure the quality of word embeddings trained on small datasets, but our novel `OddOneOut` and `Topk` tasks succeed in this regime.

London is to England as Paris is to France

can be translated perfectly fine into Middle English, but the concept of nations and capitals didn't exist in the Middle Ages, and so the analogy is not semantically meaningful. To create a meaningful analogy test set, you hire a historian fluent in Middle English, and with considerable effort and research she creates custom analogies that make sense in Middle Age England.

With this analogy test set in hand, you train dozens of models with varying hyperparameters. Unfortunately, all these models get 0 accuracy on your test set. You simply don't have enough data to get good results on the analogy task. As Figure 1 shows, the analogy task requires a large training dataset before it begins getting non-zero results. (Further details are provided in Section 3.1 below.) But that does not mean that you cannot train word embeddings on Middle English.

In this paper, we introduce the `OddOneOut` and `Topk` tasks for evaluating word vectors on low-

resource languages, and successfully train word embeddings for Middle English, 15 other extinct languages, and a low-resource emoji embedding task. To get a sense of scale, the original word2vec paper trained English word embeddings on a dataset with 6 billion tokens (Mikolov et al., 2013) and subsequent work has improved performance by training on datasets as large as 630 billion tokens (Grave et al., 2018). In this paper, the largest dataset we consider has 41 million tokens, and the smallest only 1813 tokens. We argue that different evaluation techniques are needed for datasets like ours that are more than 1000 times smaller, and Figure 1 shows that the `OddOneOut` and `Topk` tasks can measure improvement in word embedding quality with much smaller datasets than the analogy task.

Other work in the low-resource regime has focused on developing new training methods rather than evaluation methods. Specifically, the goal is to reduce the sample complexity of word embedding models by adding new regularizations (Adams et al., 2017; Jiang et al., 2018; Gupta et al., 2019; Jungmaier et al., 2020). A common thread of this work is the difficulty of evaluation. Unfortunately, each of these works evaluate their method only in a simulated low-resource environment using modern English text and not on any actual low-resource languages. They do this specifically because no evaluation metrics were available that were suitable for their low-resource target languages. More theoretical work has also shown that these simulated low-resource design methodologies give biased hyperparameter estimates which systematically overestimate model performance (Kann et al., 2019). This highlights the need for new evaluation methods like ours suitable for the low-resource regime.

High-resource languages also directly benefit from our methods in two ways. First, we help automate evaluation on many languages. Grave et al. (2018) trained FastText embeddings on 157 languages using data from the Common Crawl project. But they were only able to explicitly evaluate 10 of these language models using the analogy task due to the expense required in developing appropriate test sets. We introduce a method that automatically generates test sets for our `OddOneOut` and `Topk` methods in any of Wikidata's 581 supported languages (which includes extinct languages like Middle English).

Second, many applications of word embeddings investigate low-resource subsets of high-resource languages. There is growing body of digital humanities work where English language text is subdivided into smaller corpora based on time periods (e.g. Kulkarni et al., 2015; Hamilton et al., 2016b,a; Dubossarsky et al., 2017; Szymanski, 2017; Chen et al., 2017; Liang et al., 2018; Tang, 2018; Kutuzov et al., 2018; Kozlowski et al., 2019) or different political ideologies Azarbonyad et al. (2017). Word embeddings are then trained on these smaller corpora, and differences in the resulting embeddings are used to track how word usage changes. Our evaluation methods can be used to improve the ability to evaluate this work as well.

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. 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 using 16 extinct languages from the Classical Languages ToolKit (CLTK) library (Johnson, 2014). We evaluate these word embeddings on a qualitative downstream task that investigates how religious words are used in each language.

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 in previous work, and demonstrate the versatility of our 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 16 extinct languages.

[1] URL hidden for blind review.

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 the 16 language corpora. Section 5 concludes by discussing how extensions to this work could serve communities working with low-resource languages.

FIXME: *Figure 1: Change the x-axis to "training dataset size (number of unique tokens)"; also, the font is a bit small*

2 Evaluation Methods

The OddOneOut and Topk tasks are simple and widely applicable. Both tasks require a test set consisting of a list of categories, where each category contains a list of words that belong to that category. Table 4 shows some example categories generated through a fully automated process (described in Section 4.1). The Topk task measures a model's ability to identify words that are related to each category, and conversely the OddOneOut task measures a model's ability to identify words that are unrelated to each category,

Formally, assume that there are m categories, that each category has n words[2], that there are v total words in the vocabulary, and that the words are embedded into $\mathbb{R}^d$. The method of generating the embedding (e.g. word2vec, GloVe, FastText) does not matter. 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 $O(dvk)$.[3] 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

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

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

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 $O(dkmn^k v)$. This exponential dependence on k is very bad. In practice, we used $k = 3$ in all of our experiments, but even this small value required prohibitively long run times.

To solve this problem, we use a sampling strategy. Let $\tilde{P}$ denote the set of p samples without

[2] 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.

[3] 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.

replacement from the set P. Then we rewrite Equation 6 as

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

With this definition, the runtime of $\texttt{OddOneOut}(k)$ is $O(dkmpv)$, which is linear in all the parameters of interest. 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 using simulated English data. Second, we show that the OddOneOut and Topk metrics are useful for model selection in an emoji embedding task where the analogy task is not well defined. This experiment also demonstrates that 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.[4] 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[5], 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. We generate test sets for

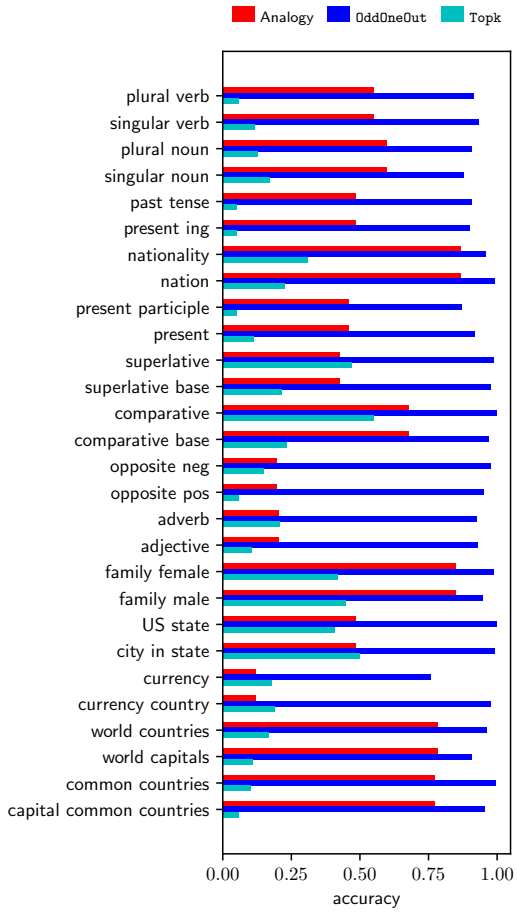

Figure 2: Breakdown of model performance by category. There does not appear to be any correlation between the performance of the three tasks, indicating that each task is measuring a different aspect of linguistic knowledge. ***FIXME:*** *the category column should be labeled as "categories" and the key should be labeled as "tasks"*

[4] 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.

[5] Embedding dimension 100, number of epochs 1, learning rate 0.025, window size is 5, min count is 5.

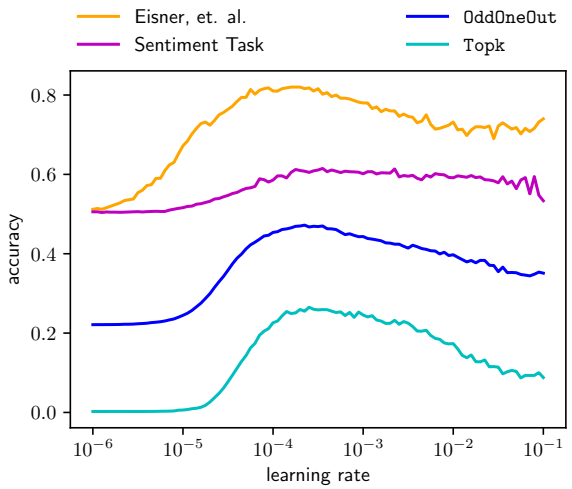

Figure 3: The performance of the generic `OddOneOut` and `Topk` tasks mirrors the performance of Eisner et al. (2016)'s emoji-specific model selection task as the learning rate varies. All three tasks provide can be used to estimate the optimal learning rate for the downstream sentiment classification task.

the `OddOneOut` and `Topk` tasks from all 28 categories in the Google test set. For example, the countries-capitals analogy set has analogies like

London is to England as Paris is to France

comparing countries and capitals. We use the set of all countries as one category and the set of all capitals as a separate category in the `OddOneOut` and `Topk` test sets. Note that in this dataset conversion, we are explicitly losing information about how these categories relate to each other. Note that the analogy task tests a model's knowledge about each category and how these categories relate to each other, whereas the `OddOneOut` and `Topk` tasks will only test a model's knowledge about each category individually. Since these tasks are testing less knowledge, it makes since that they will get improved performance on smaller data sets. This intuition is confirmed in the results shown in Figure 1. Figure 2 shows a breakdown in accuracy by category on the model trained on the full dataset. How a category performs on one task does not seem correlated to how a model performs on other tasks, which indicates that all three tasks are in fact measuring different aspects of linguistic knowledge (and not just representing the same knowledge scaled differently).

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. That is, we don't sacrifice performance by choosing our easy-to-use and widely applicable metrics over harder-to-use domain-specific metrics.

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 6 semantically meaningful 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 1661 of the 3000 Unicode emojis are included.[6] 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 the subset of 1661 emojis supported by the

[6]New emojis have been added to the Unicode standard since the publication of Eisner et al. (2016), but these emojis have not been added to the test set.

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

"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.

Note that it is incorrect to conclude that the Eisner et al. (2016) method is better than the `OddOneOut` and `Topk` values because it achieves higher accuracy rates in Figure 3. When evaluating a model selection metric, the important point to consider is the location where the metric is maximized, and not the maximum value itself. The location is used to determine the optimal hyperparamter, and all three metrics have maximal performance at the same location.

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 (Johnson, 2014).[8] Each of these languages is "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. Our techniques successfully let us choose hyperparameters for 16 of the 18 languages under consideration, including for the tiny Ancient Gujarati model.

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.

[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 embeddings.

Biblical Figures: *Abraham, Hilkiah, Matthew the Apostle, Zerubbabel, Joseph*

Facets of Buddhism: *Buddha, Vipassan, Bhagavan, meditation, nirvana*

Facets of Hinduism: *Matm, Rishi, Hindu prayer beads, Ashvadamedha, Brahmin*

Figure 4: Example categories and terms for our test set that we automatically extracted from Wikidata. Only the English translations are shown. ***FIXME:*** *include 10-15 (exactly 3 lines each) alphabetically ordered* ***FIXME:*** *are these the exact names in wikidata? I couldn't find the categories in google*

4.1 Test Set Generation with Wikidata

One of the most difficult and time consuming steps of evaluating word embeddings on a new language is generating a high quality test set for this language. We now present the first fully automatic way to generate these test sets. Our method uses Wikidata[9], which is the knowledge base that powers Wikipedia and contains millions of items and their semantic relationships. Wikidata supports 581 languages, and our test set generation method works for all of them. This method does not work for generating analogy test sets, but only works for generating the category test sets needed for our `OddOneOut` and `Topk` tasks. We implement this process using the Wikidata Query Service and SPARQL in our open source Python library, making it easy to generate arbitrary test sets.

The idea is straightforward. Some items in Wikidata actually represent categories, and other items can be an "instance of" these categories. For example, item `Q20643955` (the category of "Human Biblical Figures"), and item `Q9181` (the Biblical patriarch "Abraham") is an instance of `Q20643955`. We can then generate a category of "Human Biblical Figures" by gathering all the items that are an instance of this category, and extracting the translation of these items into our chosen language.

There are two minor complications to the process above. First, the item may not have a translation into all languages. Item `Q3276278` (the minor Hebrew prophet Hilkiah), for example, does not have a translation into any of the languages

[9] `https://wikidata.org`

Corpus			Hyperparameters (*)							Metrics		
Language	Total Tokens	Uniq. Tok.	Model	Type	Dim	WS	LR	MC	Lem	OddOneOut	Topk	Avg
Helenic												
Anc. Greek	37 868 209	1 877 574	w2v	cbow	40	9	0.1	4	False	0.190 0	0.017 8	0.032 7
Italic												
Latin	17 777 429	470 790	w2v	cbow	50	10	0.1	7	False	0.152 7	0.064 5	0.090 8
Old French	68 741	8 343	fast	sg	250	6	0.1	8	False	0.000 1	0.010 9	0.000 3
Germanic												
Mid. English	7 048 144	314 527	fast	sg	90	5	0.1	7	False	0.023 9	0.001 2	0.002 4
Mid. High German	2 090 954	60 674	fast	cbow	15	6	0.1	3	False	0.000 5	0.002 9	0.001 0
Old English	104 011	33 018	fast	cbow	425	3	0.1	3	True	0.000 0	0.000 5	0.000 2
Old Norse	458 377	59 186	w2v	cbow	60	10	0.1	3	False	0.056 9	0.009 3	0.016 1
Old Swedish	1 297 740	116 374	fast	sg	50	8	0.1	5	False	0.003 1	0.000 1	0.000 4
Indic												
Bengali	5 539	2 323	fast	cbow	15	3	0.1	4	False	0.000 0	0.006 5	0.000 2
Gujarati	1 813	1 140	fast	sg	80	3	0.1	5	False	0.000 0	0.029 2	0.000 2
Hindi	587 655	55 483	fast	cbow	45	4	0.1	8	False	0.017 5	0.003 8	0.006 4
Malayalam	9 235	5 405	-	-	-	-	-	-	-	-	-	-
Marathi	797 926	96 778	w2v	sg	400	4	0.1	6	False	0.021 4	0.006 5	0.010 1
Punjabi	1 024 075	31 343	fast	sg	50	8	0.1	5	False	0.000 0	0.000 1	0.000 1
Sanskrit	4 042 204	896 480	w2v	sg	35	9	0.1	10	False	0.139 1	0.009 3	0.017 5
Telugu	537 673	276 330	w2v	cbow	60	10	0.1	3	False	0.004 2	0.000 0	0.000 2
Semitic												
Classical Arabic	81 306	20 493	-	-	-	-	-	-	-	-	-	-
Hebrew	41 378 460	893 512	fast	sg	30	4	0.1	3	False	0.064 6	0.005 6	0.010 4

(*) w2v = word2vec, fast = fastText, sg = skipgram, Dim = Dimension, WS = Window Size, LR = Learning Rate, MC = Min Count, Lem = Lemmatization

Table 1: The optimal hyperparameters selected for training a model on each corpus, and their corresponding evaluation metrics. (Ave denotes the hyperbolic mean of OddOneOut and Topk.) We successfully trained models on 16 of the 18 languages provided by the CLTK library (everything except Malayalam and Classical Arabic). Previously, word embeddings had only been trained on Ancient Greek and Latin.

we are studying except for Hebrew. We replace all of these words with a special out-of-vocabulary token. During the evaluation tasks, the models will be guaranteed to miss any test case involving these words. This will cause the model's performance to decrease, but it will not cause the optimal set of hyperparameters to change because the model's performance will decrease uniformly for all hyperparameters. This is acceptable because our primary goal with the OddOneOut and Topk metrics is model selection.

Second, the item may have a multi-word translation. Item Q43600 is translated into English as "Mathew the Apostle" which is guaranteed to be out of vocabulary because our embeddings are only for individual words and not phrases. We handle this case simply by treating phrases the same way we would treat any other out of vocabulary word for these models. For a word2vec model, these phrases are commonly replaced by a single out of vocabulary token, but FastText models incorporate sub-word information are therefore able to generate reasonable vectors that are good representations of these phrases. We therefore expect FastText models to perform better on datasets with many multi-word translations.

In this paper we focus on three wikidata categories that we hypothesized would be relevant to the languages in CLTK we are studying: Q20643955 (Biblical Figures), ***FIXME:*** ??? (Facets of Hinduism), and ***FIXME:*** ??? (Facets of Buddhism). Table 4 shows representative examples from each category. The categories are large, with between 100-400 items each, and so we do not reproduce the full categories here.

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. Table 2 defines the hyperparameters and the range of values sampled for each. All hyperparameters are arguments to GenSim's model training functions with the exception of lemmatization. This boolean argument indicates whether we used CLTK's built-in lemmatization tools to preprocess the datasets before training with GenSim. In theory, lemmatization can improve sample efficiency by giving words with the same stem the same word embedding. The results in Table 2, however, show that

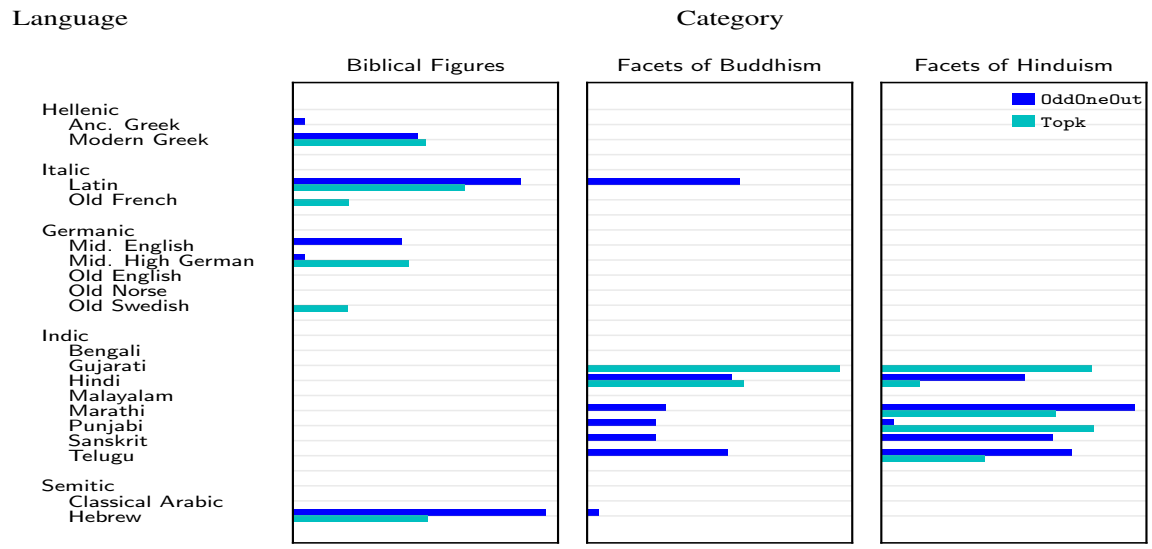

Figure 5: ***FIXME:***

this was not the case in practice, and this indicates that the lemmatization models built-in to CLTK likely have very high error rates.

To ensure a fair comparison, we randomly sampled 100 sets of hyperparameter combinations.[10] 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. In particular, we note that there is no pattern regarding whether the word2vec model is better than the fastText model, or whether the CBOW type is better than the skipgram type.

In previous work, Al-Rfou et al. (2013) trained word2vec embeddings on 100 different languages using Wikipedia as the training data and Grave et al. (2018) extended this work by training FastText embeddings on 157 languages using data from the Common Crawl project. In both cases, they researchers tuned the model's hyperparameters on only a single language (due to the difficulty of adapting an analogy test set to so many different languages), and then applied the same set of hyperparameters to all languages. Our results here suggest that performance could be improved if each model's hyperparameters were tuned individually, and our wikidata technique would make this a realistic option.

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

Hyperparameter	Sampled Range
Model	word2vec, fastText
Type	skipgram, CBOW
Dimension	5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500
Window Size	3, 4, 5, 6, 7, 8, 9, 10, 11
Learning Rate	10^{-3}, 10^{-2}, 10^{-1}
Min Count	3, 4, 5, 6, 7, 8, 9, 10, 11
Lemmatization	True, False

Table 2: The set of values sampled from for each hyperparameter during our random search hyperparameter optimization.

4.3 Language Comparison

We now breakdown the performance of each of our language models on the three religious categories shown in Figure 4. The goal is to better understand which topics are discussed in each of the ancient languages. Figure 5 shows the results. There are three interesting results in this visualization.

1. As would be expected, the European and Semitic languages get better results on the Biblical Figures categories, whereas the Indic languages get better results on the Facets of Buddhism and Facets of Hinduism categories. More surprisingly, all of the "old" European languages, however have small accuracies on the Biblical Figures category. Europe was heavily Christianized at the times all these lan-

guages were spoken, but we speculate that the church's official language being Latin causes this shortcoming.

2. Among the Indic languages, it is interesting that about half the languages get good results for both Buddhism and Hinduism, and half the languages get no results for either category. ***FIXME:*** *Is there anything more to say about this?*

3. Latin stands as an outlier in the Facets of Buddhism category as a non-Indic language. ***FIXME:*** *What else?*

5 Conclusion

We introduced the first word vector evaluation methods designed specifically for the low-resource domain, `OddOneOut` and `Topk`, and a method for automatically producing test sets in Wikidata's 581 supported languages. We believe that Wikidata is an underutilized resource in the NLP evaluation community, and in particular that it's massively multilingual support can help the NLP community better server under resourced languages.

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.

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*.

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.

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.

Jana Chamonikolasová. 2014. Middle english. *A concise history of English*, pages 46–61.

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*.

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

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*.

Kyle P. Johnson. 2014. CLTK: The Classical Language Toolkit. https://github.com/cltk/cltk.

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.

Katharina Kann, Kyunghyun Cho, and Samuel R. Bowman. 2019. Towards realistic practices in low-resource natural language processing: The development set. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3342–3349, Hong Kong, China. Association for Computational Linguistics.

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.

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

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*.

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.