Instructions for EMNLP 2020 Proceedings

Anonymous EMNLP submission

Abstract

The analogy task introduced by (Mikolov et al., 2013) has become one of the most popular methods for evaluating the quality of word embeddings; however, evidence suggests that analogies may not perform equally well across training contexts. Specifically, this method has proven non-trivial to adapt to languages beyond English and empirical evidence suggests that it requires large amounts of training data. As a result, those working in low resource or specialized settings have lacked a standard method to evaluate the quality of word embeddings. To address this problem, we propose two different intrinsic evaluation methods, which achieve greater performance sensitivity than the analogy task in low resource settings. We demonstrate the flexibility of these methods by using them to tune embeddings for 18 low resource languages as well as the full list of unicode emojis. Furthermore, our methods are designed such that custom test sets can be developed semi-automatically in over 400 different languages, which helps increase access for research in areas previously inhibited by low amounts of data.

1 Introduction

The use of high quality distributed word embeddings has become important to achieve good performance on many tasks in natural language processing. One of the first and most popular methods for generating embeddings is word2vec introduced by (Mikolov et al., 2013). However, the popularity of word2vec was not won on the idea of distributed word embeddings alone, but also due to the high quality of their embeddings as evidenced by the ability to solve analogy relationships. Famously, they used the offset between pairs of vectors to reveal 'linguistic regularities' such as `king` $-$ `man` $+$ `woman` $=$ `queen`. Using this framework, (Mikolov et al., 2013) formulated a

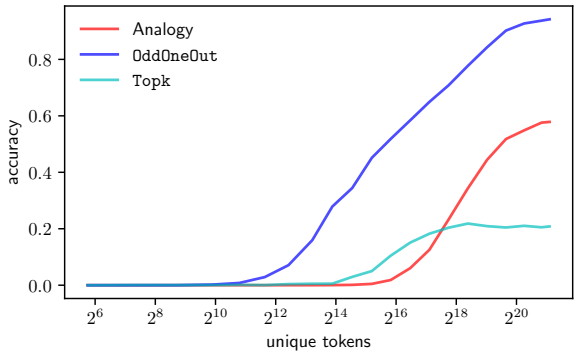

Figure 1: The plot above compares the regions of effectiveness for our evaluation metrics. The analogy task fails to measure change in accuracy of the embeddings until the number of unique words in the training dataset reaches 2^{16}, much later than both `OddOneOut` and `Topk`. Though `OddOneOut` seems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

14 category evaluation set consisting of a mix of semantic and syntactic relationships which was released alongside a set of pre-trained word vecors. Since its release, the Google analogy test set has effectively become the gold standard for measuring embedding quality and spawned much work focused on using analogies to capture other linguistic regularities, increasing performance on the analogy test set, and translating the test set to be used with non-English languages. At the same time, only small consideration has been given to areas where the analogy task either cannot be applied or fails to capture meaningful differences in quality between sets of embeddings.

Most work regarding word embeddings has been carried out in resource rich settings. In particular, the original word2vec embeddings were trained on a GoogleNews corpus containing 6 billion English tokens of which 783 million were unique (Mikolov et al., 2013). Unsurprisingly, greater amounts of

training data leads to more accurate embeddings. But in reality, there are many domains in which this amount of data is impractical or even impossible to obtain. Consider for example, a low resource language such as Telugu or a historical language like Latin. Furthermore, resource rich languages can quickly become low resource when considered from a specialized perspective. Those studying the evolution of language over time often require very specific corpora which in turn restricts the quantity of data.

More recently, embeddings have even been generated for tokens that serve the purpose of words, but are not considered words in the traditional sense. The clearest example of these non-traditional embeddings is emoji, which have emerged as an important feature of natural language data in social media. In this setting it is unclear whether the analogy task should be applied and if so how to do so systematically and at scale.

In this paper, we demonstrate that the analogy task is not always suitable for evaluating quality of word embeddings, particularly when training resources are limited and when working in niche or specialized domains. As a solution to this problem, we propose `OddOneOut` and `Topk`, two alternative evaluation methods which provide greater usability in low resource settings and are more general in nature. To further increase flexibility in deployment of embedding evaluation systems we also present Wikidata as a tool for generating test sets for these methods in up to 461 languages.

2 Evaluation Methods

From a technical standpoint, our methods are quite simple and designed to be as generally applicable as possible. Both `OddOneOut` and `Topk` require a test set which consists of groups of categories where each category contains words that are semantically similar. The `OddOneOut` method seeks to identify the word that does not belong when presented with a group of words. For a given word, `Topk` finds the k most similar words according to the model and compares them to words that belong to the semantic category from which the word was chosen.

We now formalize the methods. Assume that there are m categories, that each category has n words, that there are v 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 $\texttt{FindMostSimilar}(k, w)$ return the k most similar words in the vocabulary to w. We define the `Topk` score for class i to be

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

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 `FindMostSimilar` is $O(dvk)$.[1] 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 input data and evaluation set.

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

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

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

[1] We use gensim's implementation of `FindMostSimilar`, 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.

$|P| = O(n^k v)$. Finally, we define the k-th order `OddOneOut` score of the entire evaluation set to be

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

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.

3 Experiments

3.1 English Experiments

First we seek to illustrate how performance on the analogy task, `OddOneOut`, and `Topk` relates to size of the training corpus. Since the original GoogleNews dataset is not available to the public we instead use a 2017 dump from English Wikipedia readily available through the gensim library. This dataset consists of 2 billion tokens with 2 million unique. While not quite as large as GoogleNews, it serves as a good proxy for the high resource setting in English.

We test the relationship between amount of training data and embedding performance on the analogy task by training a series of models on subsets of our corpus. We train our first model with a subset containing 2^6 unique tokens and double the number of unique tokens for each subsequent model until all tokens are used. Since the Wikipedia corpus is alphabetized by article, a sequential streaming of the data would introduce bias. The smallest models would only be trained on content related to articles near the beginning of the alphabet. This is a problem because larger models would have the benefit of both greater amounts of data as well as a greater contextual variety of data compared to smaller ones. To mitigate this bias we use a linear congruential generator to randomly stream the training data in a way that is reproducible. Thus, in addition to mitigating model bias tied to the content of training data, our random generator also ensures that each model uses a subset of the training data of models larger than it. In total, we train 25 word2vec models with the following fixed set of hyperparameters: dimension = 100, epochs = 1, learning rate = .025, window size = 5, min count = 5.

To calculate accuracies we use the Google analogy test set to which is easily adapted for use with `OddOneOut` and `Topk` by splitting each relationship pair into two separate categories.

Figure 2 shows the results of the evaluation of these models on three intrinsic tasks– `Analogy`, `OddOneOut`, and `Topk`. We see that for the various tasks both `OddOneOut` and `Topk` start to see an increase in performance with roughly $32x$ less data compared to `Analogy`. In addition, though the shapes are roughly similar, the curve for `OddOneOut` and `Topk` have steeper slopes before plateauing, indicating a greater sensitivity to amount of data. Overall, these findings suggest that in the low resource scenarios, the analogy task does will not always provide adequate intrinsic evaluation for word embeddings, especially compared to these simpler methods.

3.2 Emoji Experiments

In addition to performing poorly as an evaluation metric in low resource circumstances, the analogy task does not easily generalize beyond traditional word embeddings. One example of this where this is prevalent is in the case of emoji embeddings. With the rise of social media data, a good amount of semantic data has been showed to be captured in the form of emojis. In 2016 (Eisner et al., 2016) introduce pretrained emoji embeddings known as emoji2vec. In their paper, (Eisner et al., 2016) mention the analogy task as a way of evaluating the quality of their embeddings; however, the analogy task is not easily adaptable to this specialized domain. As a result they were forced to settle for a qualitative analysis of small number of analogy relationships that were translatable. In order to quantitatively evaluate and tune their set of emoji embeddings, (Eisner et al., 2016) develop their own 'emoji-description classification' method which required the creation of a test set with manually labeled emotion-description pairs.

The simple and general design of our methods allows them to be easily adapted to intrinsically evaluate emoji embeddings. We demonstrate this by training our own emoji embeddings using (Eisner et al., 2016) code with `OddOneOut` and `Topk`

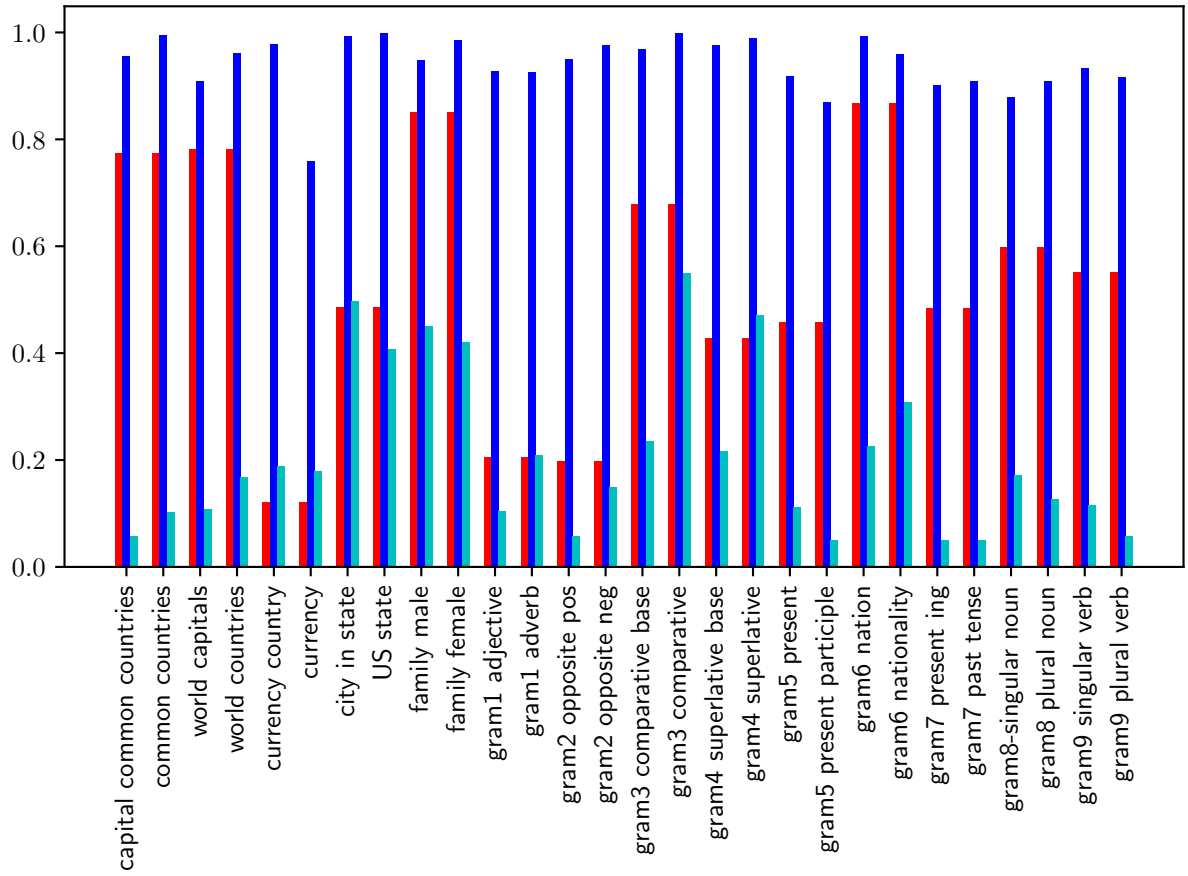

Figure 2: The methods perform better on some categories than others. `Topk` seems to excel in categories that are more homogenous like 'family female', while analogies seem to work best with geographical relationships. Note that in adapting the Google analogy set to work with our methods required splitting 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.

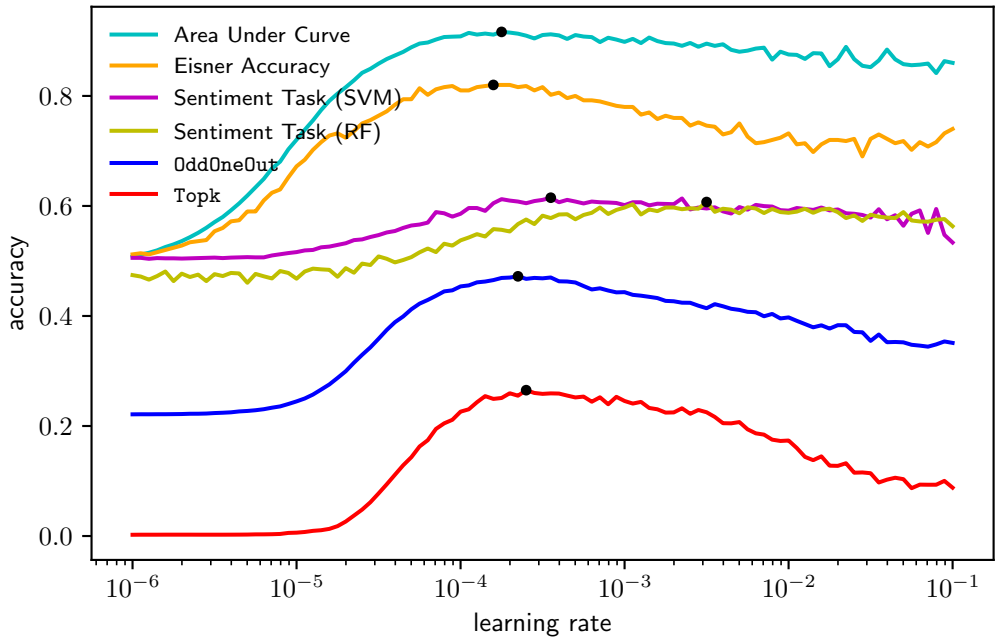

Figure 3: This figures shows the tuning of emoji embeddings across different learning learning rates where the max accuracy for each metric is marked with a point. Both `Topk` and `OddOneOut` follow the shape of (Eisner et al., 2016) area under the curve and accuracy metrics. Our methods lead us to select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

as the fine tuning metric. The semantic categories of emojis are trivial to generate from unicode's full emoji list[2]

In addition to intrinsic evaluation of emoji2vec with their custom method, (Eisner et al., 2016) also deploy their vectors in a downstream sentiment analysis of tweets. We verify the performance of our tuned emoji embeddings by also evaluating performance on this same sentiment analysis task. Figure 3 shows the results of evaluation, we find that our embeddings were able to reproduce (Eisner et al., 2016) performance on the downstream task.

3.3 Automatic 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.

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.

3.4 Classical Languages Experiments

An obvious application of `OddOneOut` and `Topk` is in training models for under-resourced languages. The Classical Language Toolkit (CLTK) (et al., 2014–2019) is one such ongoing project supporting

[2] `https://unicode.org/Public/emoji/13.0/emoji-test.txt`

	Corpus				Parameters							Scores		
	Language	Test Set	Tokens	Unique Tokens	Model	Type	Dim	Window	LR	Min Count	Lemma	OddOneOut	Topk	Combined
Hellenic	greek	ancient greek	37 868 209	1 877 574	w2v	cbow	40	9	-1	4	False	1140	8	17.86
	greek	modern greek	37 868 209	1 877 574	fast	sg	90	7	-1	5	True	300	9	19.36
Italic	latin	latin	17 777 429	470 790	w2v	cbow	50	10	-1	7	False	2748	48	96.28
	old french	french	68 741	8343	fast	sg	250	6	-1	8	False	1	7	3.20
Germanic	middle english	english (old)	7 048 144	314 527	fast	sg	90	5	-1	7	False	239	7	15.48
	middle high german	german	2 090 954	60 674	fast	cbow	15	6	-1	3	False	9	19	13.33
	old english	english (old)	104 011	33 018	fast	cbow	425	3	-1	3	True	0	1	1.33
	old norse	icelandic	458 377	59 186	w2v	cbow	60	10	-1	3	False	968	2	5.98
	old swedish	swedish	1 297 740	116 374	fast	sg	50	8	-1	5	False	50	1	3.85
Indo-Aryan	bengali	bengali	5539	2323	fast	cbow	15	3	-1	4	False	0	2	1.50
	gujarati	gujarati	1813	1140	fast	sg	80	3	-1	5	False	0	7	1.78
	hindi	hindi	587 655	55 483	fast	cbow	45	4	-1	8	False	263	2	5.93
	malayalam	malayalam	9235	5405	-	-	-	-	-	-	-	-	-	-
	marathi	marathi	797 926	96 778	w2v	sg	400	4	-1	6	False	342	1	3.98
	punjabi	punjabi	1 024 075	31 343	fast	sg	50	8	-1	5	False	0	1	1.3
	sanskrit	sanskrit	4 042 204	896 480	w2v	sg	35	9	-1	10	False	1530	1	3.99
	telugu	telugu	537 673	276 330	w2v	cbow	60	10	-1	3	False	50	0	1.96
Semitic	classical arabic	arabic	81 306	20 493	-	-	-	-	-	-	-	-	-	-
	hebrew	hebrew	41 378 460	893 512	fast	sg	30	4	-1	3	False	1098	6	13.91

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.

nlp for various classical, low resource, and often historical languages. To confirm the efficacy of our methods in the low resource setting, we train and tune word embeddings on all languages supported by the Classical Language Toolkit which contain at least one downloadable corpus and a tokenization tool. In total this leaves us with 18 low resource languages.

For each language we preprocess all corpora available through CLTK, using their normalization and tokenization tools where appropriate. We then perform a randomized grid search over the key model parameters consisting of 100 samples. The key hyperparameters we searched over can be found in Table 1. After training the model we perform an intrinsic evaluation for the embeddings on the corresponding language specific test set and pick the model that had the highest `Combined Score` which is calculated as the harmonic mean of `OddOneOut` and `Topk`.

Although the parameter combinations searched over for each language were the same, the best model for each language contained a unique set of parameters. This supports the need for an intrinsic tuning method for word embeddings so that highest performance can be achieved. The tuning results in Table 1 also seem to indicate that fastText models perform better than word2vec in low resource scenarios due to the training of n-grams.

4 Multilingual Content Analysis

Aside from intrinsic evaluation, our methods can be leveraged as a qualitative downstream task we call multilingual content analysis. The goal of the task is to determine the degree to which some concept or category is captured by a set of word embeddings. We implement this by using Wikidata to construct 3 categories of interest–Biblical Figures, Facets of Hinduism, and Facets of Buddhism–and then evaluate them using `OddOneOut` and `Topk`. The performance on this task demonstrates the degree to which our models understand the semantics of the chosen topics and provides insight regarding the content they were trained on.

With intentionally selected categories we are able draw out the differences in content between the different language models that in many cases matches our intuition. By evaluating on a category of biblical figures we see highest performance from the Hebrew and Latin models followed by other European languages; whereas Facets of Buddhism and Facets of Hinduism see high performance from the Indic language models.

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 disposal. 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.

5 Acknowledgements

6 Related Work

References

Ameeta Agrawal, Aijun An, and Manos Papagelis. 2018. Learning emotion-enriched word representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 950–961.

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

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

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

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

Mohammed Elrazzaz, Shady Elbassuoni, Khaled Shaban, and Chadi Helwe. 2017. Methodical evaluation of arabic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 454–458.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Anoop Kunchukuttan. 2020. The IndicNLP Library. https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf.

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

Stamatis Outsios, Christos Karatsalos, Konstantinos Skianis, and Michalis Vazirgiannis. 2019. Evaluation of greek word embeddings. *arXiv preprint arXiv:1904.04032*.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Rachele Sprugnoli, Marco Passarotti, and Giovanni Moretti. 2019. Vir is to moderatus as mulier is to intemperans-lemma embeddings for latin. In *CLiC-it*.

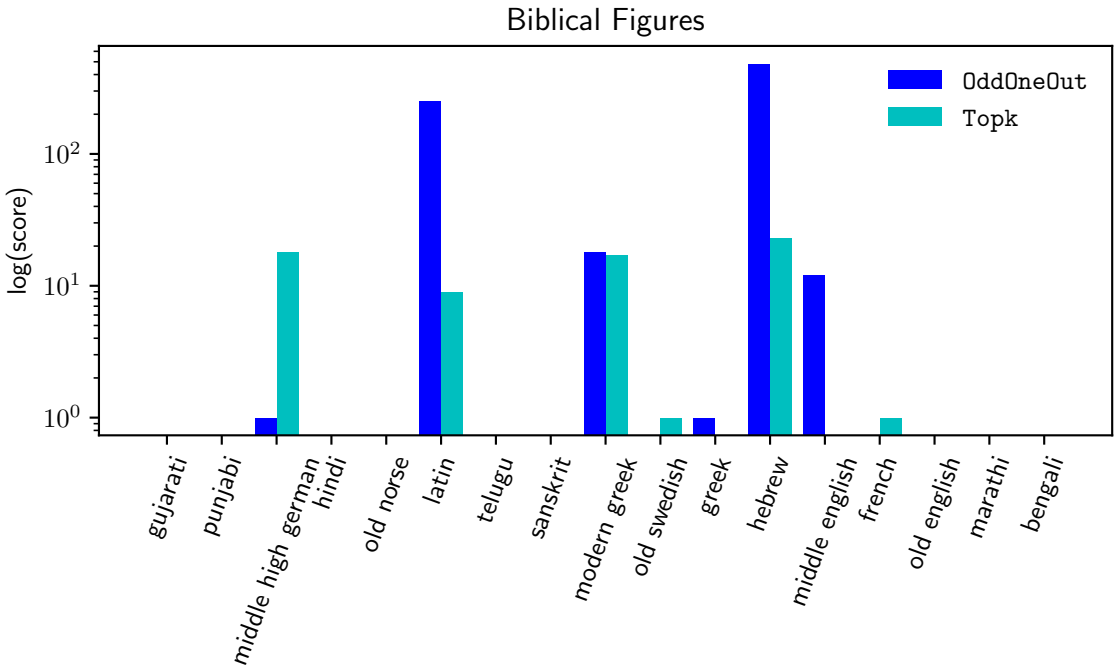

Figure 4: High performance on the Biblical Figures category indicates some level of biblical influence via the corpus. Interestingly, we see that greek embeddings optimized on the Modern Greek test set significantly outperformed the embeddings optimized for the Ancient Greek. This matches our intuition that things of a biblical nature have had a greater influence on Modern Greek than Ancient Greek.

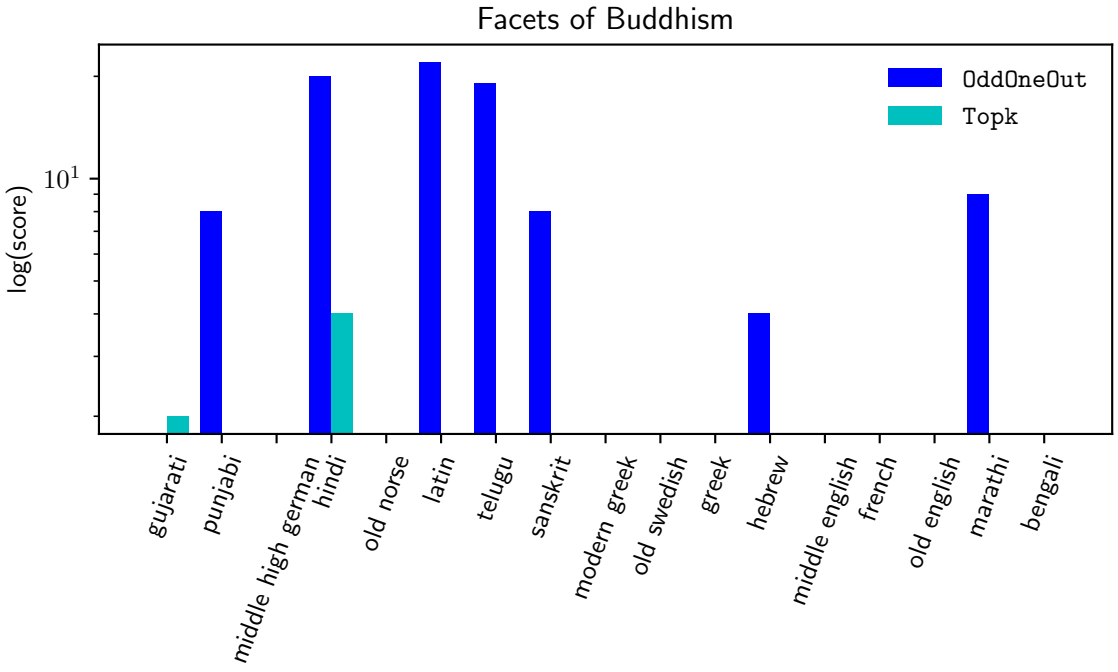

Figure 5: Though many of the Indic language embeddings performed well on the Facets of Buddhism, surprisingly so did our Latin and Hebrew embeddings. This leads us to believe that some Buddhist concepts and words are shared by corpora spanning languages as diverse as Hebrew, Latin, and Hindi. At the 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.

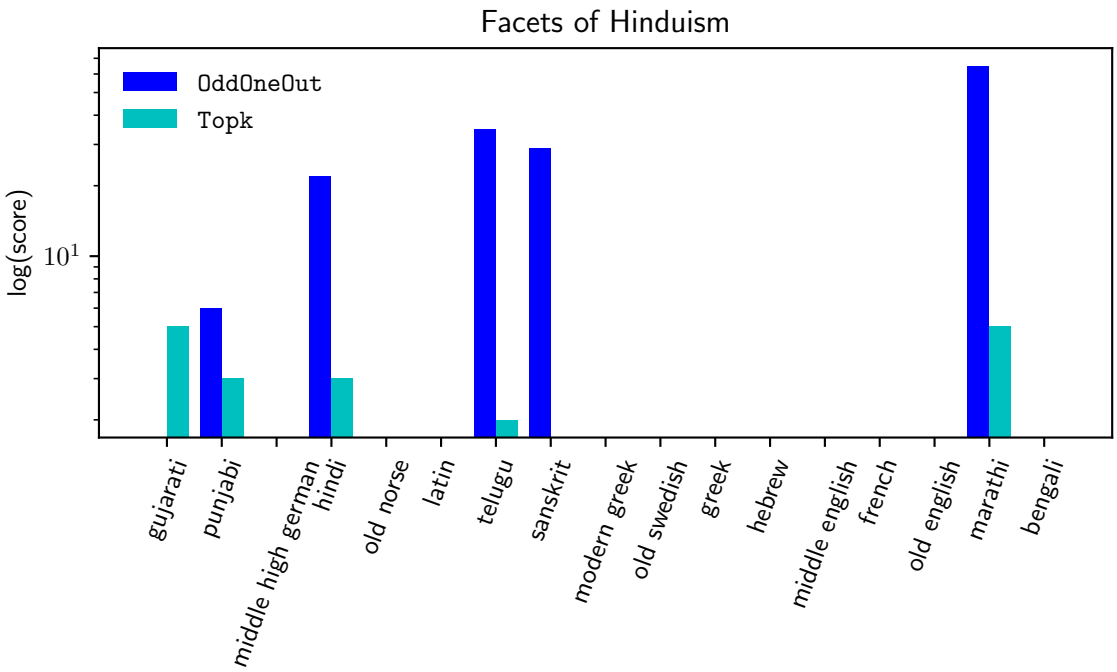

Figure 6: Similar to Figure **??** we see languages more closely related to the topic of the category achieving better performance, in this case primarily the Indic languages. Interstingly, our Bengali embeddings performed poorly on this category suggesting that the corpus did not refer heavily to the Hindu context