[RESUBMIT] Multilanguage Word Embeddings for Social Scientists:
Estimation, Inference, and Validation Resources for 157 Languages*

Pedro L. Rodriguez †
Arthur Spirling ‡
Brandon M. Stewart §
Elisa M. Wirsching ¶

Abstract

Word embeddings are now a vital resource for social science research. However, obtaining high-quality training data for non-English languages can be difficult, and fitting embeddings therein may be computationally expensive. In addition, social scientists typically want to make statistical comparisons and do hypothesis tests on embeddings, yet this is non-trivial with current approaches. We provide three new data resources designed to ameliorate the union of these issues: (1) a new version of `fastText` model embeddings; (2) a multi-language "a la carte" (ALC) embedding version of the `fastText` model; (3) a multi-language ALC embedding version of the well-known `GloVe` model. All three are fit to Wikipedia corpora. These materials are aimed at "low resource" settings where the analysts lack access to large corpora in their language of interest or to the computational resources required to produce high-quality vector representations. We make these resources available for 40 languages, along with a code pipeline for another 117 languages available from Wikipedia corpora. We extensively validate the materials via reconstruction tests and other proofs-of-concept. We also conduct human crowdworker tests for our embeddings for Arabic, French, (traditional Mandarin) Chinese, Japanese, Korean, Russian, and Spanish. Finally, we offer some advice to practitioners using our resources.

*The resources discussed in this paper can be found here: http://alcembeddings.org/

†Visiting Scholar, Center for Data Science, New York University, United States; and International Faculty, Instituto de Estudios Superiores de Administración, Venezuela, (pedro.rodriguez@nyu.edu)

‡Professor of Politics, Princeton University (as1780@princeton.edu)

§Associate Professor, Sociology and Office of Population Research, Princeton University (bms4@princeton.edu)

¶PhD candidate, Wilf Family Department of Politics, New York University (elisa.wirsching@nyu.edu)

1 Motivation

Word embeddings (e.g. Mikolov et al., 2013) are now an important tool of social science. In contrast to traditional ways of representing the contents of documents, these estimated real-valued vectors enable us to talk more directly about the 'meanings' and connotations of terms in natural language (Caliskan, Bryson and Narayanan, 2017; Rodman, 2020). Applications include modeling political emotions (e.g. Gennaro and Ash, 2022) and legislative ideology (e.g. Rheault and Cochrane, 2020). At least two challenges remain: First, obtaining high-quality embeddings for non-English languages can be difficult. Second, it has proved non-trivial to place embeddings in a modeling framework, such that one can answer questions of the form "does this group differ in a statistically significant way in terms of their embeddings of a given term"? Here, we provide resources for the union of these issues. We use the embedding models and multilingual data from the `fastText` project of Grave et al. (2018) and combine it with recent advances in "a la carte" (ALC) embeddings (Khodak et al., 2018). The latter can then be seamlessly placed in a regression-style setup courtesy of Rodriguez, Spirling and Stewart (2023).

1.1 New `fastText` Embeddings

The `fastText` project underpins the first contribution and provides two types of resources: first, an (open source) modeling architecture "that allows users to learn text representations"[1]. Second, the output of applying that embedding model to 157 languages for which training data comes from *Common Crawl* and Wikipedia. A strength of the `fastText` model is that it uses *subword* information in addition to the usual context word arrangement for prediction. This can result in higher quality embeddings than for whole words (only) because tokens that are not identical but that contain similar parts (like `policy` and `policies`) are not treated as completely separate entities. This is helpful when, say, a specific form of a word was rare in the training documents but for which we still have some information from other tokens that were more common.[2]

On inspection, we saw that *Common Crawl* includes many typos and rare terms (plus many

[1] As described here: https://fasttext.cc/

[2] See Supporting Information (SI) A for more information.

English loan words). Beyond this potential for noise, *Common Crawl* is not separated by language—it is one combined corpus that requires non-trivial division for the end-user we have in mind here. Our first contribution is simply taking the `fastText` *pipeline* and fitting it to Wikipedia in various languages. Thus, we have "our" version of `fastText`, which is cleaner than the original (though the training domain is admittedly more restricted).

1.2 New ALC Embeddings and Transformation Matrices

Our second set of contributions is to produce ALC embeddings. First, for this "new" version of `fastText`. Second, we provide ALC embeddings for `GloVe`, which we also trained on Wikipedia corpora. Details on these embeddings can be found in the SI[3] but the logic is straightforward. Essentially, embeddings of a given word w_v are estimated by taking the mean of the pre-trained embeddings of the tokens around it (u_w) and then using a *transformation matrix* (denoted $\mathbf{A}$) to redirect the new embedding away from common directions in the embeddings space (e.g., function words) otherwise likely to be over-represented in that averaging process. This allows analysts to produce high-quality vector representations even when they have very little data—including *single* instances of terms, assuming one has the context of that word and a sufficiently large corpus to pre-train embeddings. This, in turn, facilitates *statistical* inference because one can place the embeddings on "the left-hand side" and covariates of interest as predictors: for this purpose, Rodriguez, Spirling and Stewart (2023) give machinery for estimating both coefficients (on, say, group membership variables) and uncertainty around them. We provide those required (reasonable) pre-trained embeddings using both `fastText` and `GloVe` models applied to Wikipedia and the relevant learned transformation matrix. We note that while there certainly are other non-English language embedding resources (e.g. Devlin et al., 2019), they do not easily slot into a broader regression-style inference model with standard errors, p-values, etc.

[3] See SI C and the SI K.

1.3 Coverage and Intended Use

We make all required products available for 40 of the most common languages (other than English). This covers the majority of first and second-language speakers on Earth and the great majority of all languages on the web. Moreover, we have constructed pipeline production code for anyone who wishes to produce similar items for any of the 157 languages originally provided via `fastText`.

Our materials are aimed at two—often overlapping sets—of *low resource* users. First, analysts who work with languages that have relatively small corpora from which it is hard to learn high-quality embeddings. For example, scholars with a few political pamphlets or tweets from France may struggle to build embeddings for a relatively new term like "iel" (a gender-neutral pronoun) from such a small corpus. The alternative strategy—of translating the small corpus to a language for which embeddings do exist—may be unpalatable. Second, analysts who do not have local access to the computational resources required to train embedding models—we mean this both in terms of time/skill and power *per se*.

We now validate these approaches and discuss their relative performance. We first show that the ALC representations work well relative to the "full" embeddings that they approximate. We then focus high-cost efforts (i.e., crowdsourcing) on comparing (1) our version of `fastText` (fit to Wikipedia) against the original version of `fastText` and then (2) our version of `fastText` against an ALC version of our `fastText`. We do this because the `fastText` resources are the most innovative part of what we provide.

2 Performance And Validation

The resources we provide are useful to the extent that they provide reasonable representations of concepts, especially political ones. We now show that this is the case.

2.1 Reconstruction: ALC Embeddings Provide Reasonable Approximations of the "Truth"

Recall that ALC embeddings are an *approximation* to (what we might describe as) true ones, where "true" means the embeddings estimated from a vast corpus. We have the latter insofar as we can learn `fastText` or `GloVe` embeddings from, say, Wikipedia. We can then compare that truth to our estimate (our ALC embedding). We would hope that our ALC embedding can reconstruct that truth and, on average, be "close" to it rather than "far" from it. These standards are vague in an absolute sense, but they do allow us some comparison across languages. The unit of comparison here is 100 random terms per language, constrained to have a higher frequency than the median token in the corpus.[4] For each term and each language, we estimate the cosine similarity between its pre-trained embedding and its corpus-wide ALC embedding. In SI E we describe exactly how this test proceeds.

The cosine similarities by construction range between -1 and 1. If this number is 1, then the ALC embeddings (of our random terms) perfectly approximate our "true" embeddings; if they are zero or even negative, they provide a very poor approximation. In Figure 1, we report the results for all the languages we have worked with so far, including the mean (diamond) and the cosine for each of the 100 random terms (circles).

[4] We make this restriction mainly to ensure that terms are actually in the relevant language. Especially for smaller languages, lower-frequency terms are often loan words in English/other languages. In Figure 14 of SI I, we illustrate that we receive similar results with terms at the 25th percentile of the type distribution in the vocabulary for larger languages.

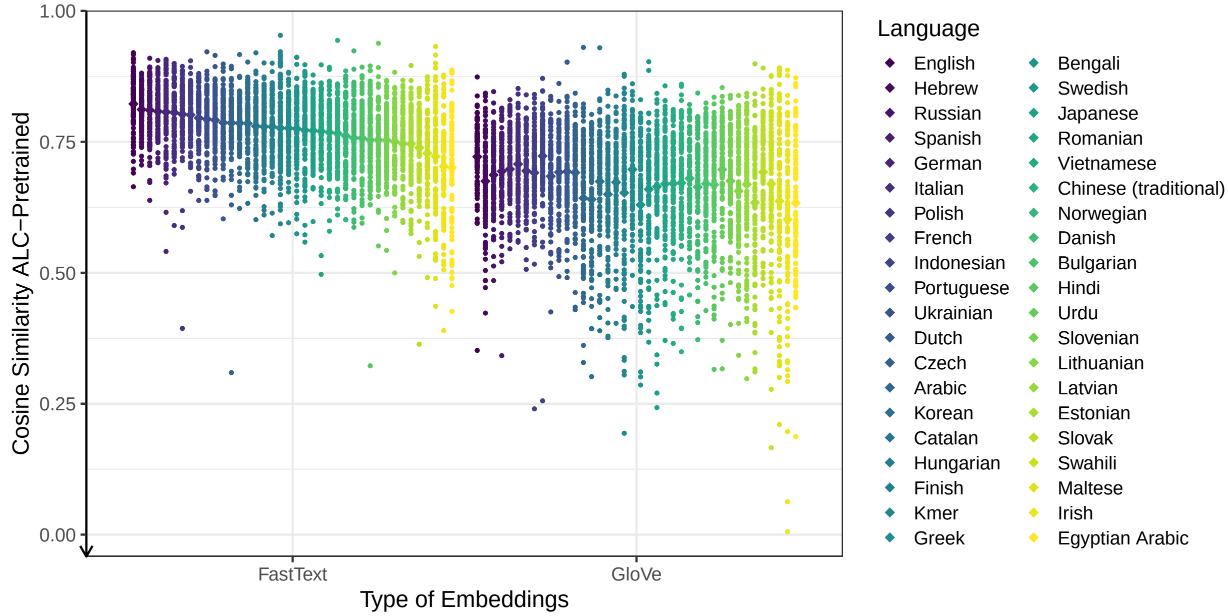


Figure 1: Reconstruction performance: cosine similarity between our ALC version of `fastText` and `GloVe` and those underlying architectures. Languages are ordered according to the mean accuracy for `fastText`. In theory, cosine similarities range between -1 and 1, but empirically all estimates are positive.

We have two immediate observations: First, ALC generally recovers both architectures' pre-trained embeddings very well for any language. In general, means are around 0.77 for `fastText` and 0.67 for `GloVe`.[5] Second, there is non-trivial variation within and between languages. In particular, and as we show more explicitly in Figure 3 of SI D, ALC does best when there is more training data—for example, English has a higher mean than Irish. Moreover, within languages with lower means, we see longer left tails—that is, there are more terms further from the mean where ALC does a worse job of approximating the "truth". Again, this is primarily a consequence of training data availability.

A more qualitatively informative procedure is to check that words represented via our embeddings "mean" what we expect them to. We first verify this by studying a curated domain setting—specifically, translated English/Spanish speeches at the European Parliament (EP), 1999–

[5] It is very difficult to make firm comments comparing within language, across models (e.g. `GloVe` v `fastText` for German). This is because the accuracy is with respect to a within-architecture baseline (`GloVe`-ALC to `GloVe`; `fastText`-ALC to `fastText`), and assumes *a priori* that the analyst seeks to model the text specifically as that architecture does.

2001 (Høyland, Sircar and Hix, 2009). We proceed as described in SI F.

2.2 Crowdsourcing: Similar Aggregate Performance, ALC Delivers More Substantive Connotations

Another and somewhat easier way to assess the quality of our embedding resources in different languages is to look at the nearest neighbors of certain political terms. Consider Table 1. There, we provide nearest neighbors (by cosine similarity) for the terms `democracy` and `equality`. The nearest neighbors are drawn from two resources: our recompiled version of `fastText` and our ALC-based version of `fastText`.[6] Consistent with our notes above, the training corpus is (English) Wikipedia.

democracy		**equality**	
our fT	our fT-ALC	our fT	our fT-ALC
democracy	democracy	equality	equality
democracy's	democratising	equalities	non-discrimination
democracies	democracy's	non-discrimination	inclusiveness
democratization	internationalism	anti-discrimination	antidiscrimination
social-democracy	parliamentarism	anti-discriminatory	anti-discrimination

Table 1: Nearest neighbors for English terms `democracy` and `equality`.

The good news is that these nearest neighbors make sense—that is, neither model produces "odd" results. Arguably, by moving beyond lexical similarities and similar word stems, ALC produces slightly more "useful" results than the pure `fastText` model. The same is true when we analyze the French terms `nationalisme` (nationalism) and `racisme` (racism), for which the training corpus is French Wikipedia, per Table 2.

[6] In Tables 5 and 6 in SI I, we repeat this exercise while further restricting nearest neighbors to terms that do not share the same word stem as the keyword. Evidently, both our `fastText` embeddings and our ALC-based version of `fastText` return meaningful nearest neighbors for political terms—beyond just lexical similarities.

nationalisme		racisme	
our fT	our fT-ALC	our fT	our fT-ALC
nationalisme	nationalisme	racisme	racisme
nationalismes	l'internationalisme	racismes	l'antiracisme
néonationalisme	internationalisme	antiracisme	communautarisme
régionalisme	radicalisme	l'antiracisme	antiracisme
internationalisme	néonationalisme	l'homophobie	l'islamophobie

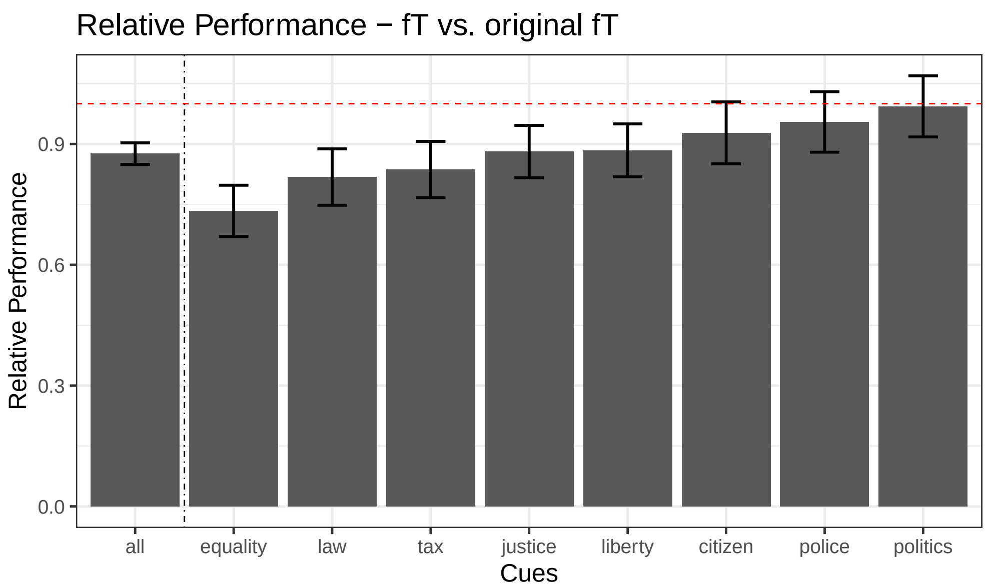
Table 2: Nearest neighbors for French terms `nation` and `racisme`.

To scale these comparisons between models, we turn to crowdsourcing (Benoit et al., 2016). Following Rodriguez and Spirling (2022), we designed a lightweight web application that shows crowdworkers a token with political connotations and then asks which of two words (drawn from two models) the worker thinks is a more plausible "context" term for that token. We translated the app into all of the (non-English) United Nations "Official Languages" and, in each language, we use eight 'political' terms (`law`, `liberty`, `equality`, `justice`, `politics`, `tax`, `citizen`, `police`). Hence, we evaluate Arabic, (traditional Mandarin) Chinese, French, Russian, and Spanish. In addition, we also created Japanese and Korean versions. If we take Rodriguez, Spirling and Stewart (2023) as sufficient evidence for the merits of ALC in English, then, combined with our exercise, we "cover" around 45% of the world's first and second languages and around 77% of the web's content languages.[7] Locating native speakers of these (non-English) languages was not trivial (and not cheap) in some cases. We worked with a specialist crowdsourcing firm, *CloudResearch*, for this purpose. In SI G we give more details on this process.

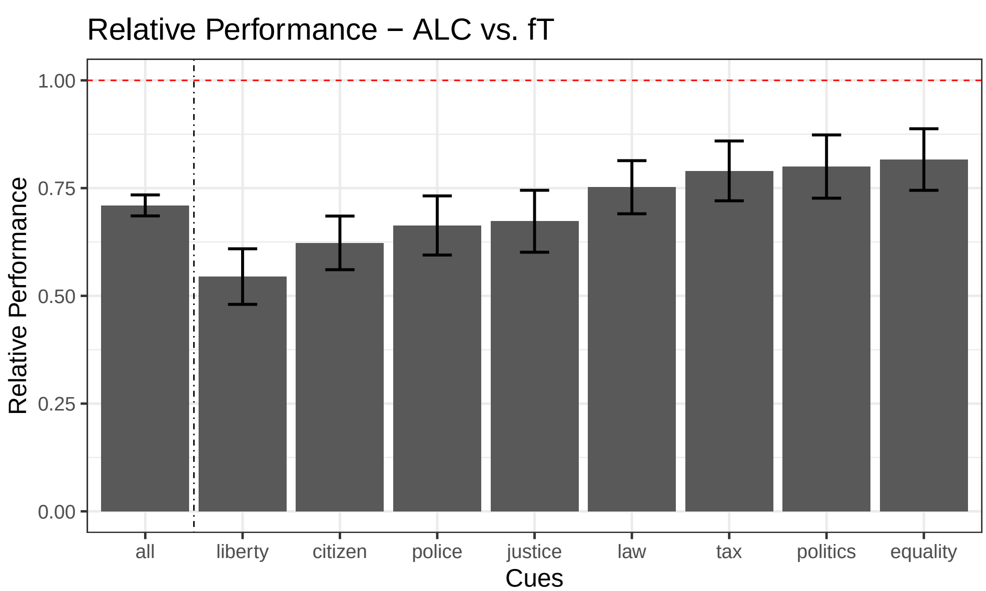
We ask crowdworkers to make two sets of comparisons: original `fastText` vs our version and then our version of `fastText` vs an ALC version of that resource. In Figure 2 we give an overview of the results. In the top subfigure, we report the comparison of our version of `fastText` to the original `fastText`. Each bar represents a term in the task (the far left bar is an overall result); we also include 95% confidence intervals. When that bar is higher than 1, respondents (on average) preferred our version; when below 1, they preferred the original. Ultimately, this comparison is

[7] See, e.g., https://w3techs.com/technologies/overview/content_language.

equivocal, with the original `fastText` being preferred in a couple of cases, but mostly, the difference is not statistically significant. The bottom subfigure compares our `fastText` to our ALC. Here, we see that, for the crowdworkers, ALC is generally not the preferred option, though again, this is equivocal in some cases.



(a) Comparing `fastText` versions



(b) Comparing `fastText` and ALC

Figure 2: Summary of crowdsourcing comparisons, all languages.
Baseline is original fT (in figure (a)) and fT (in figure (b)).

Across languages, crowdworkers mostly do not see huge differences in quality and have a mild preference for the (original) `fastText` resources (see SI H).[8] So does this mean an analyst should always prefer the original `fastText` over our version, including the one using ALC? The answer is 'no' for two reasons. First, the ALC embeddings give one access to the inferential machinery we discussed above. That is, the ALC embeddings are, by construction, an approximation, but they also allow one to conduct regressions, do statistical tests, etc. Second, and perhaps more fundamentally, these contest results disguise some important heterogeneity in use cases. Put simply, crowdworkers prefer more obvious "everyday" or "vanilla" nearest neighbors, whereas our new resources are likely helpful to analysts interested in technical terms. To see this concretely, consider Arabic—specifically, the Arabic word for `law`, قانون. The ALC nearest neighbor is المشرع (`legislator`), whereas the `fastText` nearest neighbor is قانونًيا (`legally`). Going down the list, `fastText` returns many lexical neighbors like قانونى (`legal`) and قانونه (a combination of a function word and the original keyword). Meanwhile, ALC returns more context-specific terms like الإلزام (`binding`) and التشريع (legislation).

A final note on our crowdsourcing data is that the comparisons were based on minimal preprocessing and post-processing of the embeddings. For example, we imposed only very small minimum counts for a given term to be included in their set of embeddings, specifically a minimum frequency of 10 occurrences in the language-specific Wikipedia corpus. We did this to make the comparison as 'raw' and clear as possible. However, following some internal experiments, we adjusted the various cut-offs upwards in our distributed resources. We did this especially for larger languages to ensure more robust and sensible embeddings. Put otherwise, the relative ALC vs. non-ALC crowd

[8] There is a subtlety to interpreting the results here: note that the ALC embeddings are simply averaged over the entire corpus (on which the `fastText` embeddings are themselves trained). That is, the 'context' of the ALC embeddings is the whole corpus, whereas they are actually designed, and should be optimal, for much more local use.

comparisons above are likely the worst-case scenario for ALC.[9]

3 Advice to Researchers using Our Resources

Our observations about ALC above are with reference to the relevant transformation matrix (**A**) having been estimated from the underlying corpus—specifically, Wikipedia. Unsurprisingly, whether this is appropriate for a given problem is a function of how 'close' the researcher's corpus is to Wikipedia. Here are three gradated scenarios to guide researchers in making such choices in practice:

1. Approximately in sample: if the researcher's local corpus is "close enough" to Wikipedia, then using our pre-fitted transformation matrix will work as well as anything else from the perspective of producing ALC embeddings. We demonstrate this with an example in SI J, where we use ALC embeddings for the German Wikipedia to identify homonyms.

2. Out of sample, small corpus. The researcher is out of sample if their corpus does not particularly resemble Wikipedia. If their corpus is too small to fit local models, we recommend using our estimated **A** matrix and carefully checking its validity. We give an example for this case using French and Italian parliamentary corpora in SI K.

3. Out of sample, large corpus. If their corpus is large, we advise researchers to simply fit a local transformation matrix using our pipeline code—and potentially fit their own embeddings. Of course, this involves a judgment call: the user must decide whether their inferences are better with our **A** for the language and corpus at stake or with their own (and/or with their own local embeddings). We did local fitting of **A** to our various parliamentary corpora to provide calibration. As illustrated in Appendix K, the results are satisfactory for the *Congressional Record* (median speech length 215 words) but unsatisfactory for the French and Italian corpora (median speech lengths 40 and 140 words, respectively).

[9] To reiterate, we provide full pipeline code such that users can recreate the resources under any pre or post-processing regime they wish.

To the extent researchers seek more concrete advice, our evidence suggests using our estimated quantities as a first cut on the problem. If they seem suitable and can be validated—for example, via substantive inspection of the nearest neighbors—then one can build out from there. If they do not seem suitable, consider estimating your own with our code. Subsumed in this recommendation is the idea that one might train with something other than Wikipedia on quality grounds. That is, we acknowledge that this resource has some plausible heterogeneity across languages, and analysts should use their expert judgment in deciding whether it is appropriate for their use case. In any case, our resources are a reasonable comparison point for any such work.

Acknowledgements

First version: March 17, 2023. This version: December 29, 2023. A previous version of our work won the "Best Virtual Poster" award at the Summer Polmeth Meeting (2022). Christopher Lucas, Sebastian Popa, Clara Suong, and Luwei Ying provided very helpful comments on an earlier draft. We thank Mikhail Khodak for providing us with helpful feedback. We received excellent research assistance and advice from Dias Akhmetbekov, Alia ElKattan, Tatsuya Koyama, Cristina Mac Gregor Vanegas, Francis William Touola Meda, Yinxuan Wang, and Kyu Sik Yang.

Data Availability

The resources discussed in this paper are here: http://alcembeddings.org/. This includes replication code and data.

References

Benoit, Kenneth, Drew Conway, Benjamin E Lauderdale, Michael Laver and Slava Mikhaylov. 2016. "Crowd-sourced text analysis: Reproducible and agile production of political data." *American Political Science Review* 110(2):278–295.

Caliskan, Aylin, Joanna J. Bryson and Arvind Narayanan. 2017. "Semantics derived automatically from language corpora contain human-like biases." *Science* 356(6334):183–186.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics*.

Gennaro, Gloria and Elliott Ash. 2022. "Emotion and reason in political language." *The Economic Journal* 132(643):1037–1059.

Grave, Edouard, Piotr Bojanowski, Prakhar Gupta, Armand Joulin and Tomás Mikolov. 2018. "Learning Word Vectors for 157 Languages." *CoRR* abs/1802.06893.

Høyland, Bjørn, Indraneel Sircar and Simon Hix. 2009. "Forum section: an automated database of the european parliament." *European Union Politics* 10(1):143–152.

Khodak, Mikhail, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart and Sanjeev Arora. 2018. "A La Carte Embedding: Cheap but Effective Induction of Semantic Feature Vectors." *CoRR* abs/1805.05388.

Mikolov, Tomás, Ilya Sutskever, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. "Distributed Representations of Words and Phrases and their Compositionality." *CoRR* abs/1310.4546.

Rheault, Ludovic and Christopher Cochrane. 2020. "Word Embeddings for the Analysis of Ideological Placement in Parliamentary Corpora." *Political Analysis* 28(1):112–133.

Rodman, Emma. 2020. "A Timely Intervention: Tracking the Changing Meanings of Political Concepts with Word Vectors." *Political Analysis* 28(1):87–111.

Rodriguez, Pedro L and Arthur Spirling. 2022. "Word embeddings: What works, what doesn't, and how to tell the difference for applied research." *The Journal of Politics* 84(1):101–115.

Rodriguez, Pedro L, Arthur Spirling and Brandon M Stewart. 2023. "Embedding Regression: Models for Context-Specific Description and Inference." *American Political Science Review* pp. 1–20.

Online Supporting Information:

Multilanguage Word Embeddings for Social Scientists: Estimation, Inference, and Validation Resources for 157 Languages

Contents (Appendix)

A Why fastText? 2

B GloVe embeddings 3

C ALC embeddings 3

D Details on Training Process 4
- D.1 Wikipedia Corpora 4
- D.2 Preprocessing Choices 6
- D.3 Training of fastText Embeddings 7
- D.4 Training of GloVe Embeddings 7
- D.5 Training of ALC Embeddings 7

E Reconstruction Tests: Full Description 8

F English-Spanish "translation" at the European Parliament 8

G Multilanguage Crowdsourcing Details 10

H Full Crowdsourcing Results: Model v Model 14
- H.1 Arabic 14
- H.2 Chinese (Mandarin) 15
- H.3 French 16
- H.4 Russian 17
- H.5 Spanish 18
- H.6 Japanese 19
- H.7 Korean 20

I Robustness of Validation Tests 21

J Approximately In Sample 22

K Out of Sample, "Small" Corpus 23

A Why `fastText`?

A strength of the `fastText` model is that it uses *subword* information when producing the embedding for a particular term (Grave et al., 2018). Rather than learning a single embedding vector for each whole word, `fastText` represents each word as a sum of the vector representations of its component parts. In practice, those components are n-gram contiguous characters, with special handling of word boundaries. This allows the technique to incorporate information about the internal structure of tokens.

Take, for example, a word like `policies`, and suppose $n = 3$. Then `fastText` would learn an embedding for `pol`, `oli`, `lic`, `ici`, `cie`, `ies` and one for `policies` itself. In addition, it would learn an embedding for the start and end of the word in the text, demarcating these as $<$ and $>$. That is, it also learns an embedding for <`po` and `es`>. A word is then represented by taking the *sum* of the vectors for each subword (plus the word itself and boundaries). In languages like Chinese, where words may be constructed of multiple logograms (themselves representing words), those individual characters are embedded and combined.

This can result in better predictions (and thus higher quality embeddings) because words that are not identical but that contain similar parts (like `policy` and `policies`) are not treated as completely separate entities. This is helpful when, say, a specific form of a word was rare (in the limit, absent) in the training documents but for which we still have some information from other more common tokens. This is also partly why the `fastText` technique is preferable to simply embedding the stems of terms. One can imagine that certain words with special meanings like "abortion" in US politics should not have the same representation as "abortive", even though their stem—"abort"—may be identical in some cases.

B `GloVe` embeddings

Global Vectors for Word Representation, known as `GloVe` (Pennington, Socher and Manning, 2014), are based on counts of word occurrence—that is, the frequency with which one word appears in a given window with (all) other words in a vocabulary. The co-occurrence count between words i and j is re-expressed as a probability of co-occurrence for i and j. Then the word vectors for i and j—the embeddings—are estimated such that multiplying them together (their dot product) comes as close as possible to reproducing that (log) probability of co-occurrence. This is done for all i and j, with some upweighting of probabilities where one has more data (i.e., where the underlying counts are higher). The technique can be applied to a local corpus and uses matrix factorization. Alternatively, users can download pre-existing `GloVe` embeddings fit to other corpora. These word vectors are then the *pre-trained* word embeddings for what follows.

C ALC embeddings

In the ALC settings, embeddings are derived from the additive information of pre-trained word embeddings (such as `GloVe`) in the context windows around the target word. However, simply averaging embeddings of context words over-emphasizes common words (e.g. "stop" words) (Khodak et al., 2018; Rodriguez, Spirling and Stewart, 2023). To produce "good" word representations, i.e. to recover existing word vectors $\mathbf{v_w}$, one would therefore want to rotate away from such common components by multiplying the simple additive composition of embeddings $\mathbf{u}_w$ with a "transformation matrix" $\mathbf{A}$.

$$\mathbf{v_w} \approx \mathbf{A}\mathbf{u_w} = \mathbf{A}\left(\frac{1}{|\mathbf{C_w}|}\sum_{c\in\mathbf{C_w}}\sum_{\mathbf{w'}\in c}\mathbf{v_{w'}}\right) \tag{1}$$

with the set of contexts $\mathbf{C_w}$ for word w, contexts c and context word embeddings $\mathbf{v_{w'}}$. This yields an *approximation* to the "true" embeddings of the terms of interest but allows for high-quality "local" representations of terms in the relevant embedding space. In principle, this weighting matrix can be learned from the data by minimizing the error between existing word vectors (locally trained or relying on large pre-trained corpora) and their respective additive context embeddings:

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \sum_{w=1}^{W} \alpha(n_w)\|\mathbf{v}_w - \mathbf{A}\mathbf{u}_w\|_2^2 \tag{2}$$

Here $\alpha(n_w)$ weights up words (embeddings) that are more common in the corpus and about which we have more information. This equation is a simple linear regression problem, and learning the transformation matrix is not particularly hard. What makes it difficult in practice is obtaining data on which to estimate $\mathbf{A}$. We use Wikipedia for this purpose and thus provide the transformation matrix for every language we processed so far.

D Details on Training Process

D.1 Wikipedia Corpora

As the largest free online encyclopedia, available in more than 200 languages, Wikipedia provides an important resource for multilanguage natural language processing. Importantly, because the articles are curated, the underlying text corpora are of high quality, and the corpora ensure broad coverage in terms of topics and content. We downloaded the XML Wikipedia dumps for each language[10], using the latest month available at the time of the respective download.[11] Table 3 depicts the size of the Wikipedia corpora by language. Evidently, the size of the training corpora used for our resources vary substantially and—as Figure 3 indicates—the ability of ALC to reconstruct the underlying pre-trained `fastText` embeddings correlates positively with the size of the training corpora across languages. Especially scholars working in languages with smaller Wikipedia corpora should, therefore, carefully assess the fit of our resources and compare them against a local fit of ALC embeddings.

[10] https://dumps.wikimedia.org/

[11] See the code pipeline for the specific corpora used.

Language	Number of tokens	Number of types
English (en)	2509107528	487548
German (de)	909529231	741082
Japanese (ja)	822043573	168054
French (fr)	814072051	393310
Spanish (es)	708130887	331604
Russian (ru)	544851977	556247
Italian (it)	521266262	303043
Mandarin (zh)	415856806	131437
Portuguese (pt)	307958805	279710
Dutch (nl)	280699206	269485
Ukrainian (uk)	252927316	331524
Polish (pl)	252015897	316233
Catalan (ca)	230276780	247556
Swedish (sv)	198491360	257251
Arabic (ar)	191722613	322748
Hebrew (he)	145225021	284314
Vietnamese (vi)	142363557	182936
Czech (cs)	141914459	314574
Hungarian (hu)	117150915	325888
Indonesian (id)	105720582	108170
Norwegian (no)	102522025	204184
Finish (fi)	86744273	325253
Korean (ko)	85540639	368452
Greek (el)	84863862	217338
Romanian (ro)	79237177	197309
Bulgarian (bg)	67922614	197136
Danish (da)	58681996	188904
Egyptian Arabic (arz)	48966691	151983
Slovenian (sl)	42813035	185169
Bengali (bn)	39265385	133407
Hindi (hi)	35477207	88619
Slovakian (sk)	34544078	178530
Estonian (et)	33762935	210219
Urdu (ur)	31543933	65331
Lithuanian (lt)	26653065	151937
Latvian (lv)	18603697	109113
Swahili (sw)	6909049	32927
Irish (ga)	6271469	55815
Kmer (km)	5782880	37318
Maltese (mt)	2509660	44685

Table 3: Size of Wikipedia corpora by language. All corpora have been preprocessed according to our guidelines in Appendix D.2.

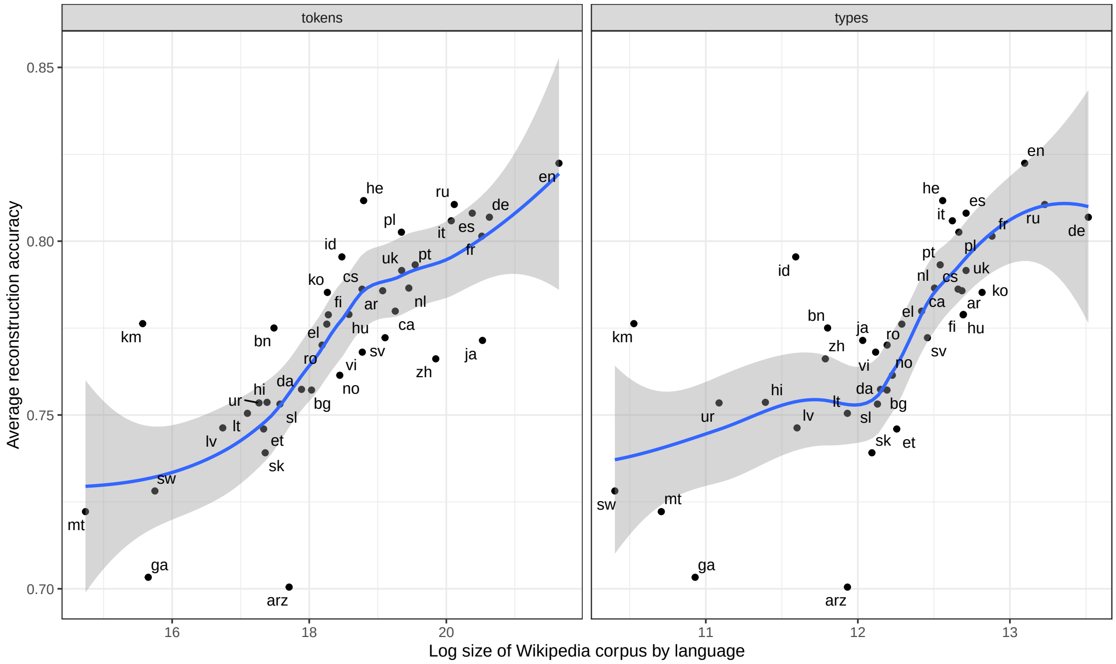


Figure 3: Relationship between size of Wikipedia corpus (number of tokens or types) and reconstruction accuracy as shown in Figure 1.

D.2 Preprocessing Choices

The first preprocessing step is to extract the text content from the XML dumps. For this purpose, we follow the `fastText` pipeline and use the `WikiExtractor` from apertium[12]. In a second step, we implement light preprocessing of the resulting corpora. In particular, we removed punctuation (except for punctuation within tokens), removed extra white space, and set all characters to lower case. Finally, we tokenize the raw text. As before, we largely follow choices made by the original `fastText` (Grave et al., 2018) to ensure better comparability of our models with existing options. We use the Stanford word segmenter for Chinese (Chang, Galley and Manning, 2008) and Mecab for Japanese (Kudo, 2005). For languages written using the Latin, Cyrillic, Hebrew or Greek scripts, we use no separate tokenizer, but split based on white space. For all remaining languages, we use the ICU tokenizer (Rui, 2020).

Additionally, when training the models (`fastText`, `GloVe` and their respective ALC embeddings), we apply a hard minimal frequency threshold for the respective vocabulary. This helps to clean out noisy parts of the corpus and thus significantly improves the fit of all models. We base our choice on the language-specific threshold on the size of the Wikipedia corpora and vocabulary by language[13]. Specifically, we impose a minimal frequency cutoff of 50 for English, 25 for medium-sized languages (i.e., German, Spanish, Italian, French, Russian, Swedish, and Dutch), 15 for small-to-medium-sized languages (i.e., Czech, Finish, Hungarian, Portuguese) and 10 for all smaller languages. As this step turned out to be crucial for the out-of-sample performance of our quantities, scholars who use our code pipeline to train resources from Wikipedia for their language

[12]https://github.com/apertium/WikiExtractor

[13]https://meta.wikimedia.org/wiki/List_of_Wikipedias

might want to experiment with the size of the threshold in their particular case.

Note that the crowdsourcing validation in the main text was done with a previous version of our resources. Following internal experiments and out-of-sample performance tests, we adjusted pre-processing and training choices after our crowdsourcing survey. Table 4 details the exact differences across the two iterations of our resources.

Category	Current Resources (as of June 2023)	Previous Resources (Crowdsourcing)
Preprocessing	• Remove punctuation btw tokens (i.e. emulate `quanteda`) • Remove extra white space • Set characters to lower case	• Remove *all* punctuation • Remove extra white space • Set characters to lower case • Remove numbers
`GloVe` training	• Vector size: 300 • Window size: 5 • Vocab min count: language-specific • x_{max} in weighting: 100 • Maximum iterations: 50	• Vector size: 300 • Window size: 5 • Vocab min count: 5 • x_{max} in weighting: 10 • Maximum iterations: 10
`fastText` training	• Skip-gram model • Vector size: 300 • Window size: 5 • Vocab min count: language-specific • Negative sampling: 10	• CBOW model • Vector size: 300 • Window size: 5 • Vocab min count: 5 • Negative sampling: 10
ALC	• Vocab min count: language-specific	• Vocab min count: 10

Table 4: Changes in training procedure across iterations of ALC resources.

D.3 Training of `fastText` Embeddings

Next, we train **`fastText`** models (Grave et al., 2018) for this preprocessed and tokenized text using a context window of 5 and setting the dimensions of the word vectors to 300. For the dictionary, we impose the minimal frequency of occurrences in the entire corpus described in Section D.2 and use negative sampling of size 10.

D.4 Training of `GloVe` Embeddings

Similarly, we train **`GloVe`** (Pennington, Socher and Manning, 2014) to our cleaned corpora. Again, we set a language-specific minimal word frequency described in Section D.2, a vector size of 300, and a context size of 5. We further impose similar parameters as in Pennington, Socher and Manning (2014), i.e., we set $x_max = 100, \alpha = 3/4$ and a maximum iteration of 50.

D.5 Training of ALC Embeddings

Finally, for both **`fastText`** and **`GloVe`** embeddings, we then train ALC embeddings (Khodak et al., 2018; Rodriguez, Spirling and Stewart, 2023) to obtain the relevant transformation matrices. We use a chunk-based learning approach to handle the large size of the respective corpora. That is, we read in the relevant preprocessed corpus by chunk and perform the following operations by chunk:

1. Retain vocabulary with a minimum term frequency of the language-specific threshold

2. Create a feature-cooccurrence-matrix (FCM) using `conText`[14], with a window size of 5 and equal weighting

3. Obtain a corresponding feature-embedding-matrix that provides additive context-specific feature embeddings (i.e., the $\mathbf{u_w}$ in Equation (1)), averaged over all embedding instances in a given chunk

To obtain the untransformed additive embeddings for all features across the *entire* corpus, we then simply average the chunk-specific additive embeddings for each feature across the chunks. This is possible because the additive context embeddings from step 3 are themselves averages of the respective instance-specific additive context embeddings in a given chunk. We do this for all features appearing with a frequency of at least the language-specific threshold across the entire corpus. Finally, we train the corresponding transformation matrix following Equation (2), where we use $\log(n_w)$ for $\alpha(n_w)$.

E Reconstruction Tests: Full Description

To fix ideas, suppose we are working with Spanish and have Spanish Wikipedia as our large, pre-training corpus ($\sim$ 639 million tokens, $\sim$ 850 thousand types). We proceed as follows:

1. Draw 100 random terms from the corpus. The only requirement on these terms is that they have higher frequency counts than the median token in the corpus.

2. Putting those 100 terms aside, produce embeddings for the large corpus via our cleaned version of `fastText` and `GloVe`. Thus we have two sets of "true" embeddings.

3. Estimate an $\mathbf{A}$ matrix in the usual ALC way for both architectures' embeddings.

4. For the 100 held-out terms for both architectures, produce an ALC embedding for each term.

   (a) For any given random term, say `pulpo` (Spanish for `octopus`), we now have an ALC embedding from `fastText` and from `GloVe`.

   (b) Calculate the cosine similarity between our ALC embedding of `pulpo` from `fastText` and the "true" fastText embedding; calculate the cosine similarity between our ALC embedding of `pulpo` from `GloVe` and the "true" `GloVe` embedding.

5. Repeat steps 4a and 4b for all of the 100 random held out words. Calculate the mean cosine distance from the "true" embeddings.

F English-Spanish "translation" at the European Parliament

We want to check that words represented via our embeddings "mean" what we expect them to. We verify this by studying a curated domain setting—specifically, translated English/Spanish speeches at the European Parliament (EP), 1999–2001 (Høyland, Sircar and Hix, 2009). To summarize: first, we produce an 'English' corpus of speeches either originally in English or translated from

[14]https://github.com/prodriguezsosa/conText

Spanish to English. Then, we produce a 'Spanish' corpus of speeches either originally in Spanish or translated from English to Spanish.

More specifically, for the English and Spanish speech data in Høyland, Sircar and Hix (2009), we proceed as follows:

1. Gather all speeches originally in English in the EP (denote as `en_orig`), and obtain their (expert) translation to Spanish (`en_to_es`).

2. Gather all speeches originally in Spanish in the EP (`es_orig`), and obtain their (expert) translation to English (`es_to_en`).

3. Combine `en_orig` and `es_to_en` into one English corpus. Use ALC to obtain the nearest neighbors of the word `but`. Compare the cosine similarity ratio ($\frac{\texttt{en\_orig}}{\texttt{es\_to\_en}}$) for each nearest neighbor to `but`.

4. Combine `es_orig` and `en_to_es` into one Spanish corpus. Use ALC to obtain the nearest neighbors of the word `pero`. Compare the cosine similarity ratio ($\frac{\texttt{en\_to\_es}}{\texttt{es\_orig}}$) for each nearest neighbor to `pero` (the Spanish translation of `but`).

The results of this exercise for the two corpora are displayed in Figure 4. We use different plotting figures to denote whether the nearest neighbor in question is from the Spanish corpus only, shared between the corpora, or from the English corpus only. To understand the figure, start with the left panel—the combined English corpora. If we assume that (a) politicians whose native languages differ (English or Spanish) do not use `but` in systematically different ways and (b) that translation is noiseless (perfect), then we would anticipate that the cosine ratio for the nearest neighbors will be 1. That is, we anticipate that, say, a term like `because` (its embedding) will be as close to `but` in the original English corpus as in the *translated to* English corpus. This is, in fact, what we see. Furthermore, we see it for all the top 10 nearest neighbors. Turning to the right part of the plot, and with evidence in hand that assumptions (a) and (b) hold from the left panel, we would hope that the nearest neighbors for `pero` will also line up at 1. If they do, we have evidence that ALC "works" for Spanish—that is, it produces reasonable nearest neighbors for terms we might care about, with which professional translators would concur. This is precisely what we see.

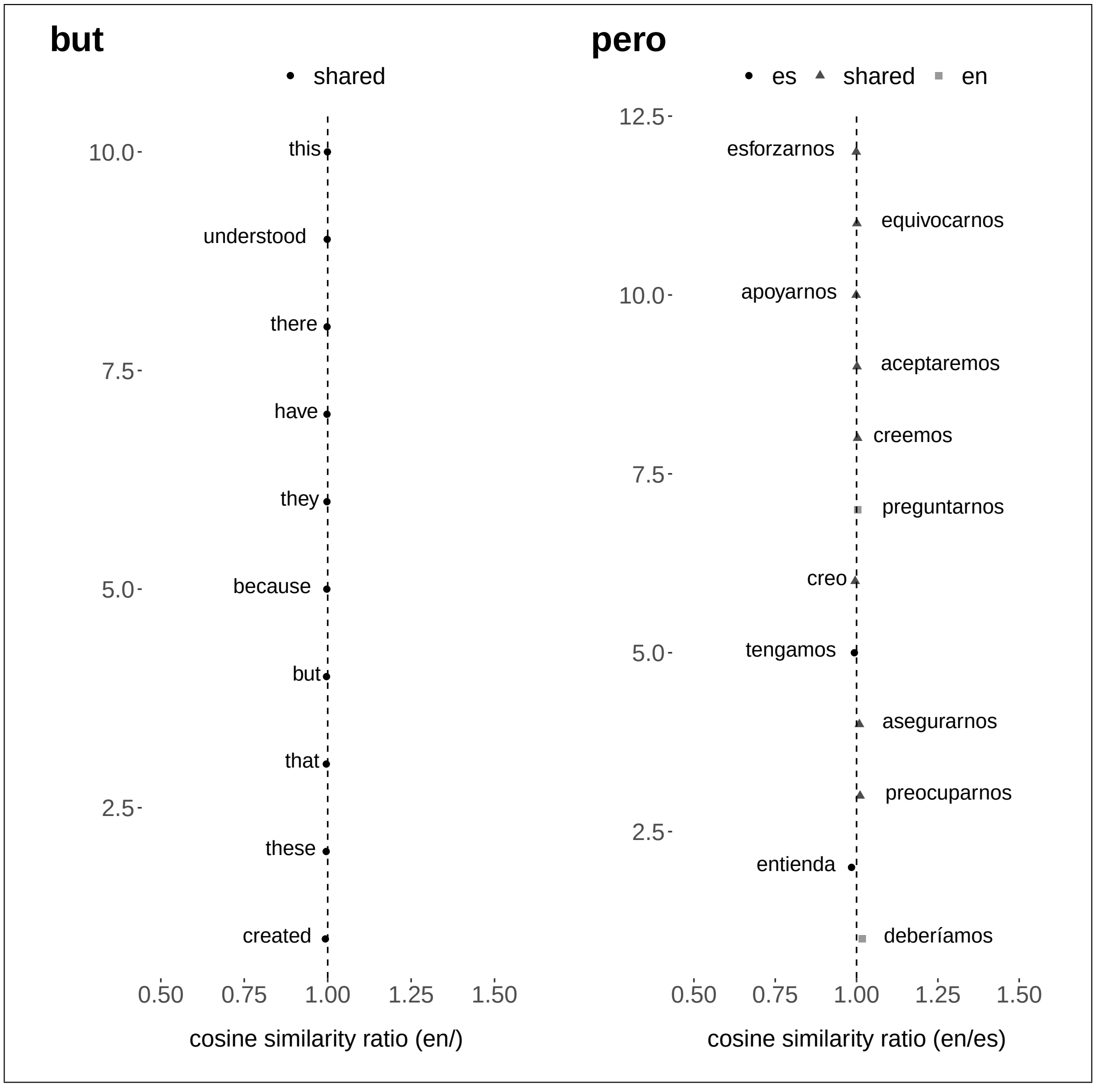


Figure 4: Translation Exercise: Cosine Similarity Ratio for ALC Nearest Neighbors is almost always 1 for translated and original texts in English and Spanish.

Notice that the embeddings themselves are *not* being translated between English and Spanish. Indeed, a feature of our multilanguage representations is that they inhabit different spaces (one per language). Our point here is that a technique (ALC) we believe works for English also works for other languages (in this case, Spanish).

G Multilanguage Crowdsourcing Details

For the crowdsourcing validation of our resources, we first employ the three embedding models we aim to compare (the original `fastText` embedding model from (Grave et al., 2018), our `fastText` model trained on Wikipedia and our ALC model using our `fastText` model for the underlying pre-trained embeddings) to obtain the top 20 nearest neighbors in the seven relevant languages for the eight political keywords (`law`, `liberty`, `equality`, `justice`, `politics`, `tax`, `citizen`, `police`) using our Wikipedia corpora. Following Rodriguez and Spirling (2022), we build a simple app that prompts crowdworkers to compare the nearest neighbors for these models. After a short introduction of the task and the general idea of keywords and context words, we ask crowdworkers to indicate which model produces nearest neighbors that best meet the definition of a context word (Figure 5 shows the task description in English). For this, we use pairwise comparisons, i.e., a given crowdworker either compares (1) the original `fastText` model to our `fastText` model or (2) our ALC model to our `fastText` model. Instead of showing the crowdworkers *all* nearest neighbors for a given keyword across the two models in the comparison, we randomly select a nearest neighbor from each set of the 20 nearest neighbors. To rule out ties, we also remove draws where nearest neighbors are identical across the two models in the comparison. We then translate the app for all relevant seven languages with the help of native speakers. Figure 6 shows an example of a pairwise comparison for `police` in Japanese and `tax` in Russian. In collaboration with *CloudResearch*[15], we then field these apps in the following regions, recruiting 50 crowdworkers for each language:

- Arabic: Saudi Arabia, Egypt, Algeria
- Chinese (traditional): Taiwan
- French: France, Canada (Quebec)
- Japanese: Japan
- Korean: South Korea
- Spanish: Mexico, Costa Rica, Colombia
- Russian: Russia, Belarus

[15]https://www.cloudresearch.com/

Context Words

A famous maxim in the study of linguistics states that:

You shall know a word by the company it keeps. (Firth, 1957)

This task is designed to help us understand the nature of the "company" that words "keep": that is, their CONTEXT.

Specifically, for a CUE WORD, its CONTEXT WORDS include words that:

- Tend to occur in the vicinity of the CUE WORD. That is, they are words that appear close to the CUE WORD in written or spoken language.

AND/OR

- Tend to occur in similar situations to the CUE WORD in spoken and written language. That is, they are words that regularly appear with other words that are closely related to the CUE WORD.

For example, CONTEXT WORDS for the cue word COFFEE include:

1. *cup* (tends to occur in the vicinity of COFFEE).
2. *tea* (tends to occur in similar situations to COFFEE, for example when discussing drinks).

Click "Next" to continue

Next

(a) General Introduction

Task Description

For each iteration of the task (13 in total including trial and screener tasks):

1. You will be given a cue word (top center of the screen) and two candidate context words (on either side of the cue word).
2. Please select the candidate context word that you find best meets the definition of a context word.
3. We are especially interested in context words likely to appear in **political discourse.**
4. If both are reasonable context words, please select whichever you find most intuitive.
5. You must select **one and only one** of the two candidate context words.

Keep in mind, some iterations are for screening purposes. These are tasks for which there is clearly a correct answer.

Wrong answers in these screening tasks will automatically end your participation so **be sure to read carefully.**

The trial task that follows is meant for you to practice. Like screening tasks, the trial task has a correct answer.

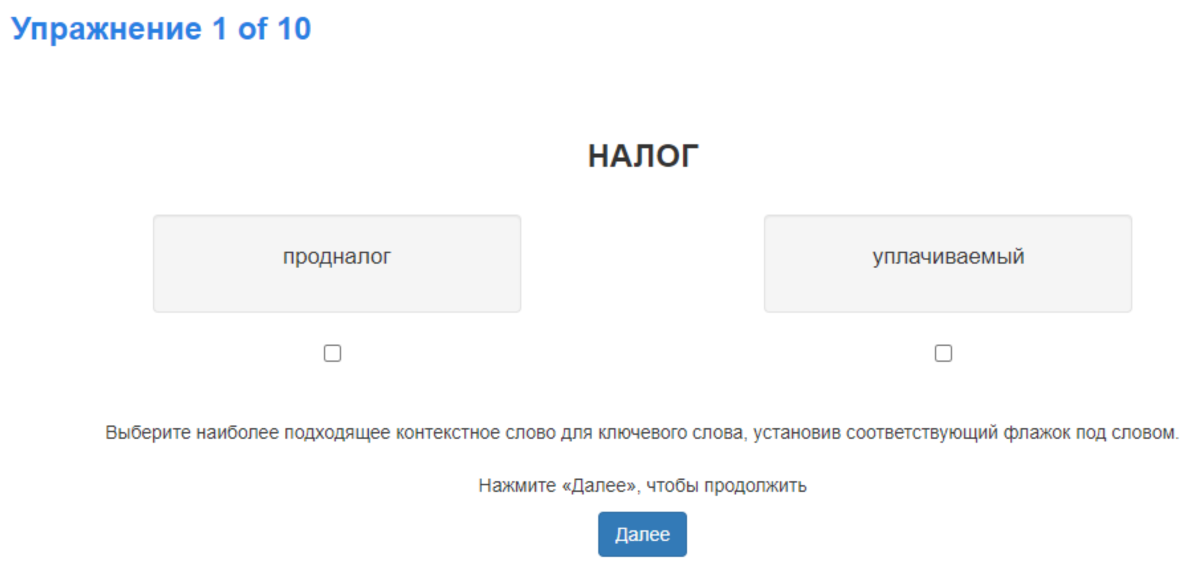
Click "Next" to continue to the trial runs

Next

(b) Task Description

Figure 5: Crowdsourcing Instructions

(a) Example of Pairwise Comparison in Japanese

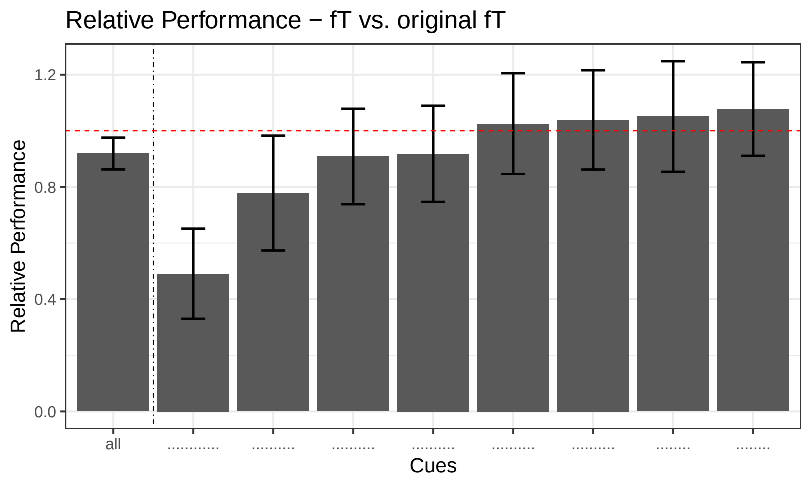


(b) Example of Pairwise Comparison in Russian

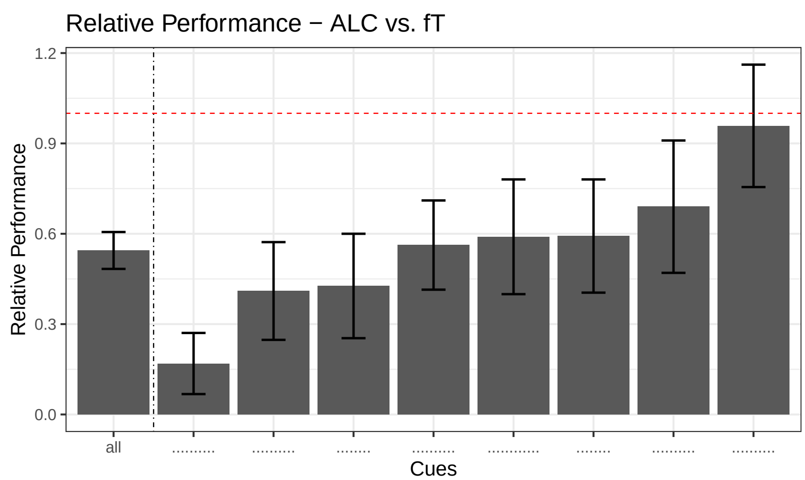
Figure 6: Crowdsourcing Examples

H Full Crowdsourcing Results: Model v Model

H.1 Arabic



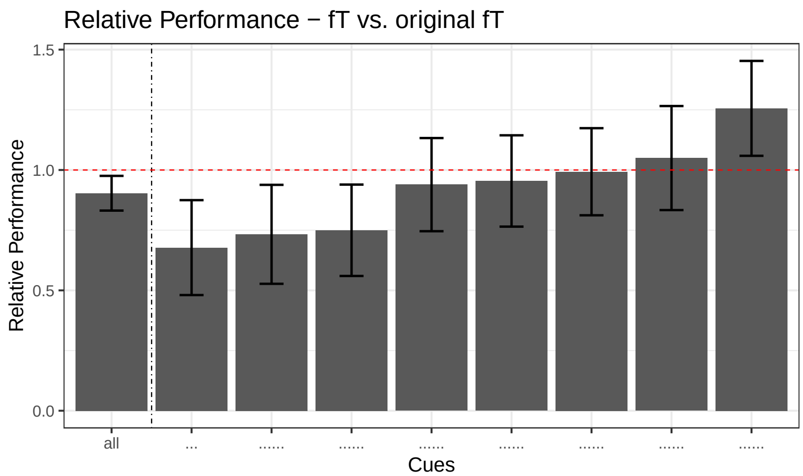
(a) Comparing `fastText` versions for Arabic



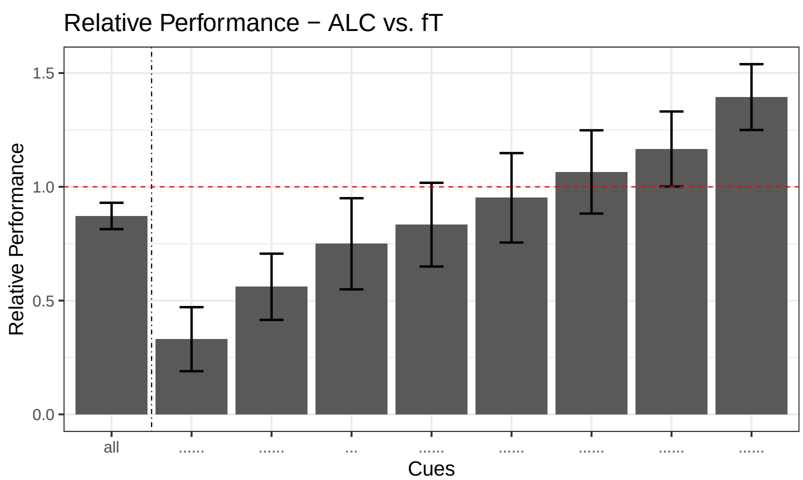
(b) Comparing `fastText` and ALC for

Figure 7: Summary of crowdsourcing comparisons for Arabic.

H.2 Chinese (Mandarin)



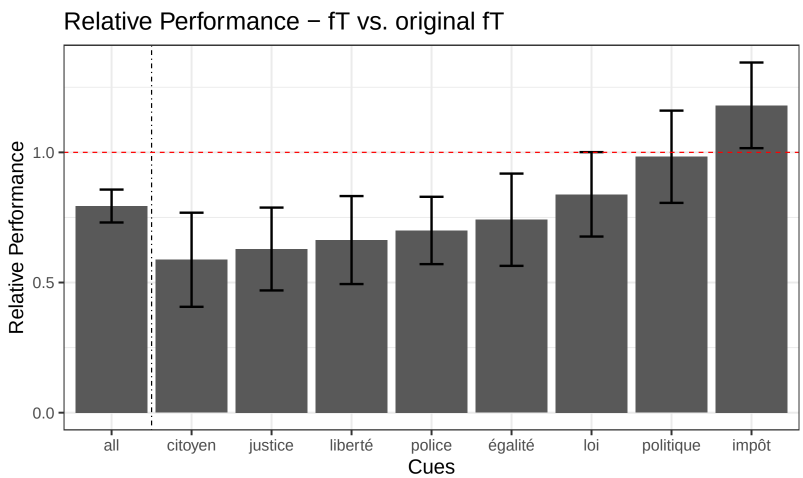
(a) Comparing `fastText` versions for Chinese



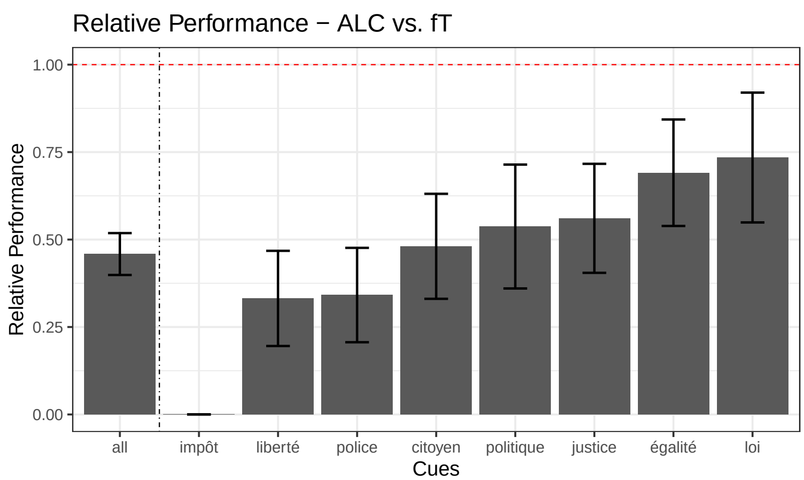
(b) Comparing `fastText` and ALC for Chinese

Figure 8: Summary of crowdsourcing comparisons for Chinese (Mandarin)

H.3 French



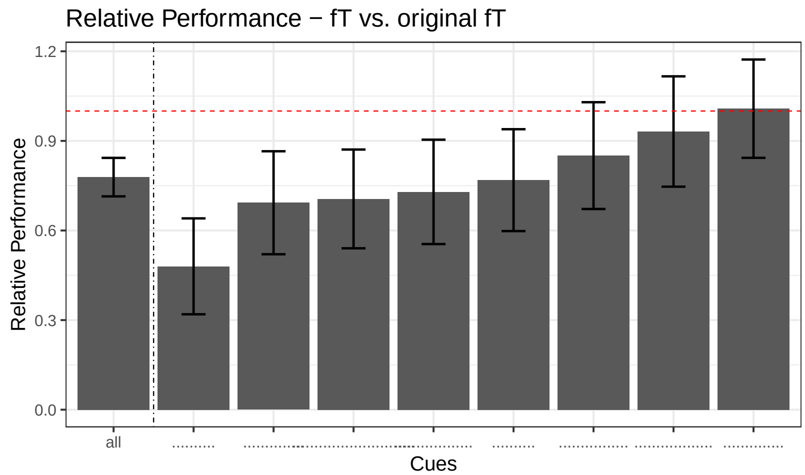
(a) Comparing `fastText` versions for French



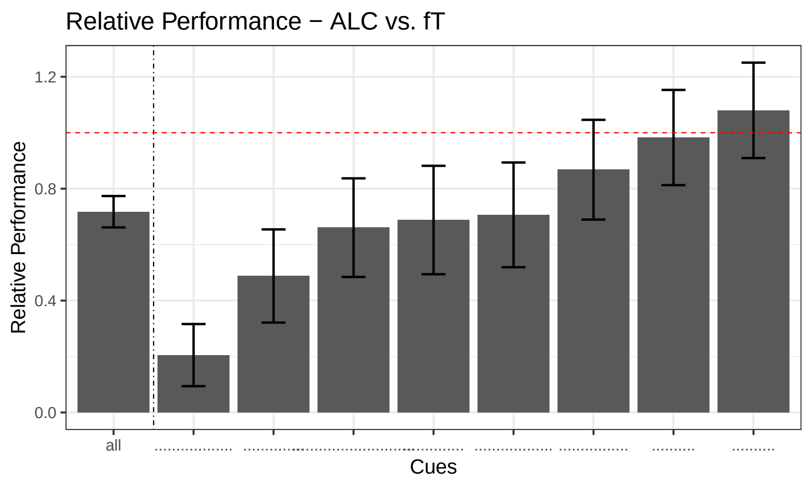
(b) Comparing `fastText` and ALC for French

Figure 9: Summary of crowdsourcing comparisons for French.

H.4 Russian



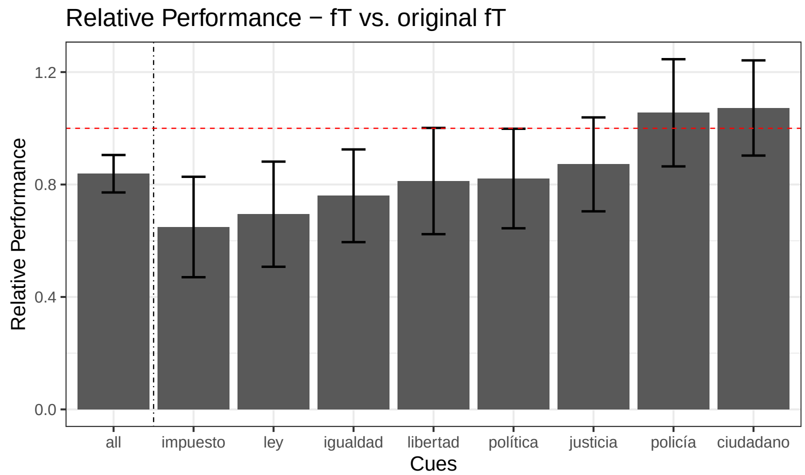
(a) Comparing `fastText` versions for Russian



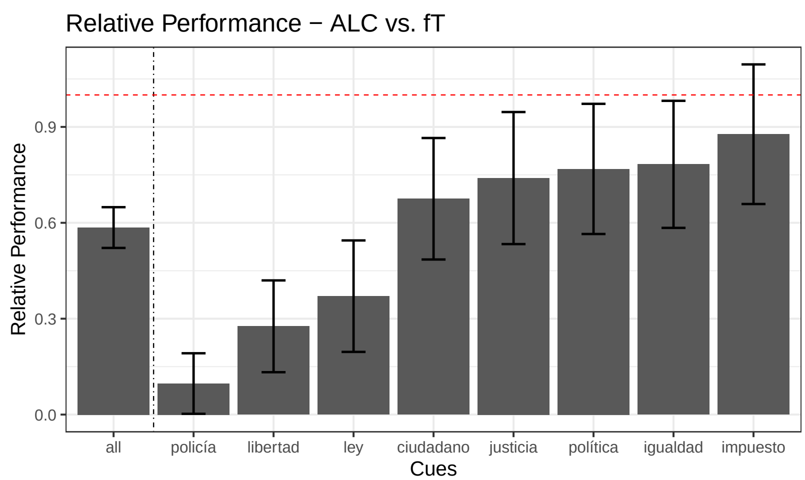
(b) Comparing `fastText` and ALC for

Figure 10: Summary of crowdsourcing comparisons for Russian.

H.5 Spanish



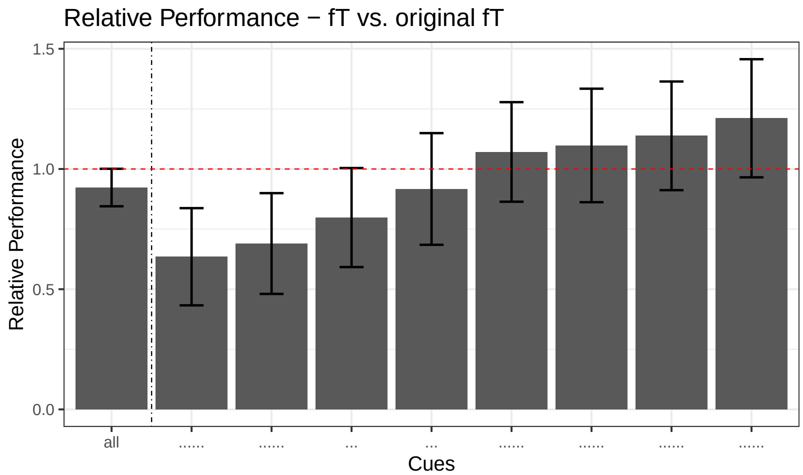
(a) Comparing `fastText` versions for Spanish



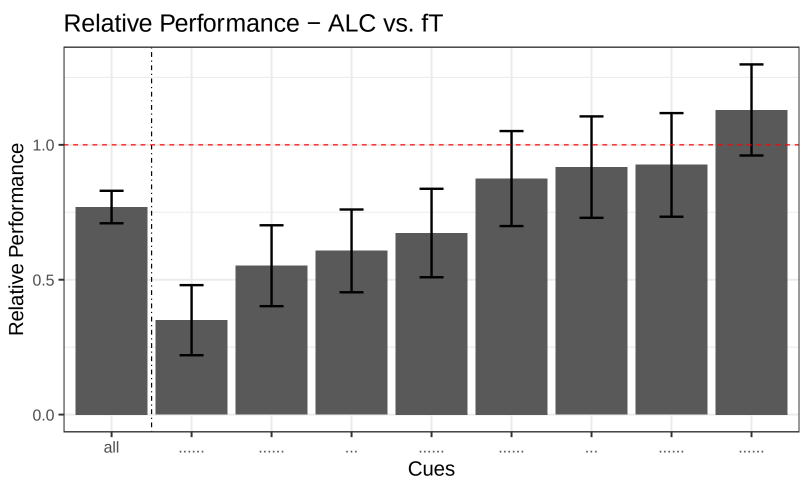
(b) Comparing `fastText` and ALC for Spanish

Figure 11: Summary of crowdsourcing comparisons for Spanish.

H.6 Japanese



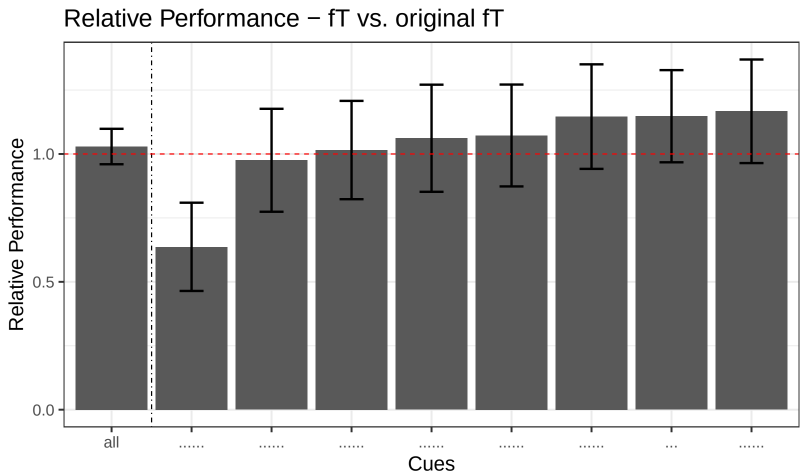
(a) Comparing `fastText` versions for Japanese



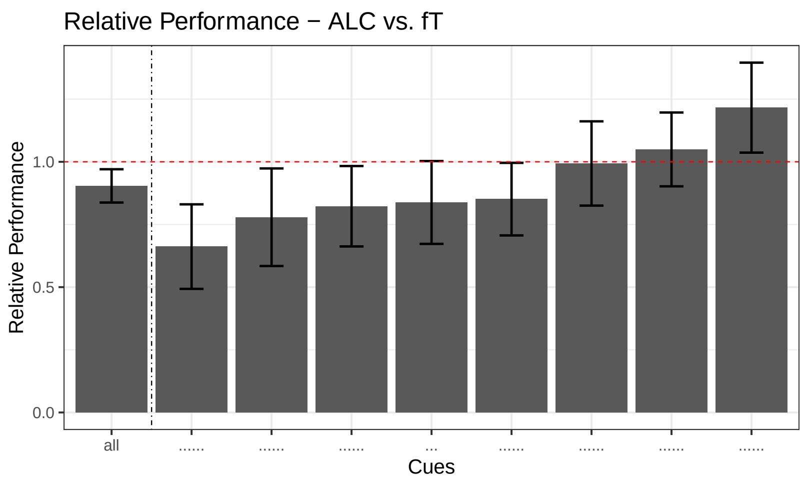
(b) Comparing `fastText` and ALC for Japanese

Figure 12: Summary of crowdsourcing comparisons for Japanese.

H.7 Korean



(a) Comparing `fastText` versions for Korean



(b) Comparing `fastText` and ALC for Korean

Figure 13: Summary of crowdsourcing comparisons for Korean.

I Robustness of Validation Tests

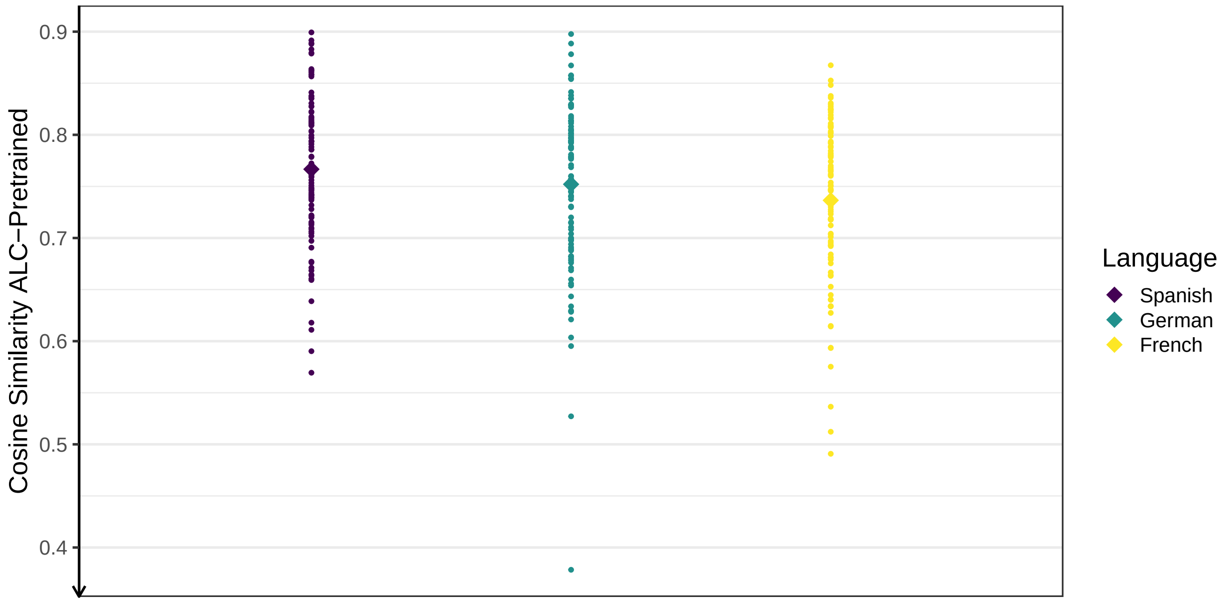


Figure 14: Reconstruction performance: cosine similarity between our ALC version of `fastText` those underlying architectures for 100 random terms at the 25th percentile of the type distribution. Languages are ordered according to the mean accuracy for `fastText`. In theory, cosine similarities range between −1 and 1, but empirically all estimates are positive.

democracy		**equality**	
our fT	our fT-ALC	our fT	our fT-ALC
democratization	democratising	non-discrimination	non-discrimination
social-democracy	internationalism	anti-discrimination	inclusiveness
e-democracy	parliamentarism	anti-discriminatory	antidiscrimination
socialism	constitutionalism	inclusiveness	anti-discrimination
democratising	democratisation	inequality	anti-discriminatory

Table 5: Nearest neighbors for English terms `democracy` and `equality`.

nationalisme		**racisme**	
our fT	our fT-ALC	our fT	our fT-ALC
néonationalisme	l'internationalisme	antiracisme	l'antiracisme
régionalisme	internationalisme	l'antiracisme	communautarisme
internationalisme	radicalisme	l'homophobie	antiracisme
l'internationalisme	néonationalisme	xénophobie	l'islamophobie
traditionalisme	progressisme	sexisme	d'islamophobie

Table 6: Nearest neighbors for French terms `nation` and `racisme`.

J Approximately In Sample

Suppose the researcher's local corpus is "close enough" to Wikipedia. In that case, using our pre-fitted transformation matrix will work as well as anything else from the perspective of producing ALC embeddings. Inevitably, there is ambiguity in "close enough", but one way to diagnose whether this is true is to, e.g., inspect the nearest neighbors and compare them to the researcher's substantive priors.

To give an example of a limiting case (i.e., being as close as possible to the training data), we illustrate the capacity of ALC to identify homonyms. These terms have identical spelling across contexts but different meanings. For instance, the German term `kiefer` means both `pine` and `jaw`, and the term `erde` can imply both `Planet Earth` and `soil`. If ALC works well with corpora that are close to or identical to Wikipedia, we would expect the context-specific embeddings to uncover these differences in meaning across contexts. As Table (7) and Figure (15) indicate, this is the case. We embed each instance of the terms `kiefer` and `erde` *à la carte* by applying our `fastText` quantities (pre-trained embeddings and transformation matrix) to the German Wikipedia. We cluster these ALC embeddings using k-means (for $k = 2$). Table (7) shows the nearest neighbors to the center of each cluster. Evidently, the first cluster of `kiefer` contains terms related to teeth and jaw bone, while the second cluster only includes other tree species, such as larch (`lärche`) or spruce (`fichte`). Similarly, the first cluster of `erde` captures terms such as vegetation cover (`planzendecke`) and rocks (`gesteinsbrocken`). In contrast, the second cluster is most closely related to words relevant to planet, sun, or moon. Given these patterns, it is not surprising that these ALC clusters are also well-separated in two principal component dimensions (Figure (15))—note the homogeneity of the word senses, with relatively little overlap on the first dimension.

kiefer		**erde**	
Cluster 1	Cluster 2	Cluster 1	Cluster 2
scherengebiss	fichten	erde	himmelskörpers
praemaxillare	nadelbaumarten	erdklumpen	magnetosphäre
protraktil	waldkiefer	erdnester	planetenoberfläche
oberkieferknochen	schwarzkiefer	gesteinsbrocken	sonnenoberfläche
kieferknochen	lärche	waldboden	sonnennähe
pharyngealia	weißtanne	pflanzendecke	meteoroiden
schläfenbein	weymouth-kiefer	lufthülle	sonnensystems
zwischenkieferbein	douglasie	vegetationsdecke	erde-mond
oberkiefers	weiß-tanne	menschenhand	äquatorebene
gaumenbein	balsam-tanne	menschenwelt	himmelskörpern

Table 7: Nearest neighbors to ALC clusters of German homonyms `kiefer` and `erde`.

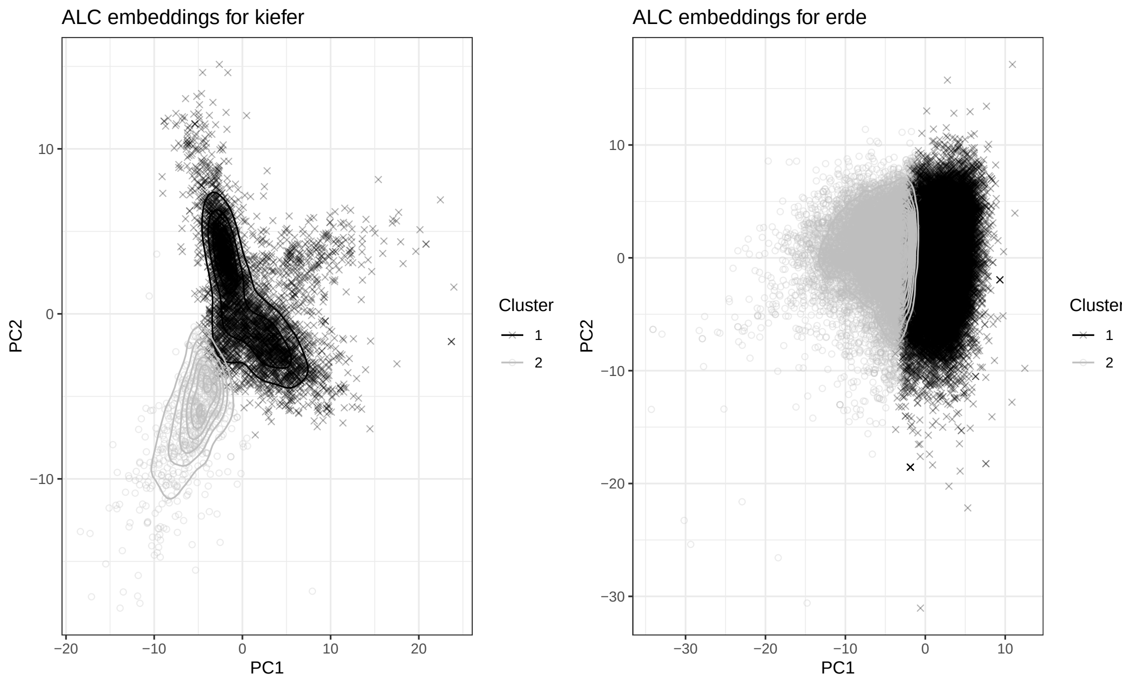


Figure 15: Identification of clusters in German homonyms `kiefer` (pine, jaw) and `erde` (Earth, soil).

K Out of Sample, "Small" Corpus

We use parliamentary corpora from the French and Italian parliamentary debates, as compiled by the *ParlaMint* project (Erjavec et al., 2023). In both examples, we use our pre-trained `fastText` embeddings together with the corresponding transformation matrices trained on Wikipedia. The first example uses the parliamentary minutes from the French National Assembly (*Assemblée Nationale*) for 2019-2020, which yields a corpus of about 216,000 documents. We show how ALC can capture changes in the meaning of certain keywords over time, specifically, how the connotation of `liberty` changes in French parliamentary debates before and after the Covid-19 outbreak. Figure (16) shows the average cosine similarity between ALC embeddings for `liberté` and our `fastText` pre-trained embeddings for relevant terms, including `pluralisme` (`pluralism`), `urgence` (`emergency`) and `visite` (`visit`). As one would expect, the figure shows how the usual nearest neighbors of `liberté`, i.e., `pluralisme`, `équité` and `discrimination`, experience a sharp drop in their cosine similarity with the ALC embedding of `liberté`. In contrast, *a priori* less closely related terms, such as `covid`, `urgence` and `visite`, show a substantially larger cosine similarity with `liberté` once the virus became a major health crisis in France. These dynamics were particularly stark in April 2020, when the Covid cases reached their first peak and the French government enacted a strict lockdown.

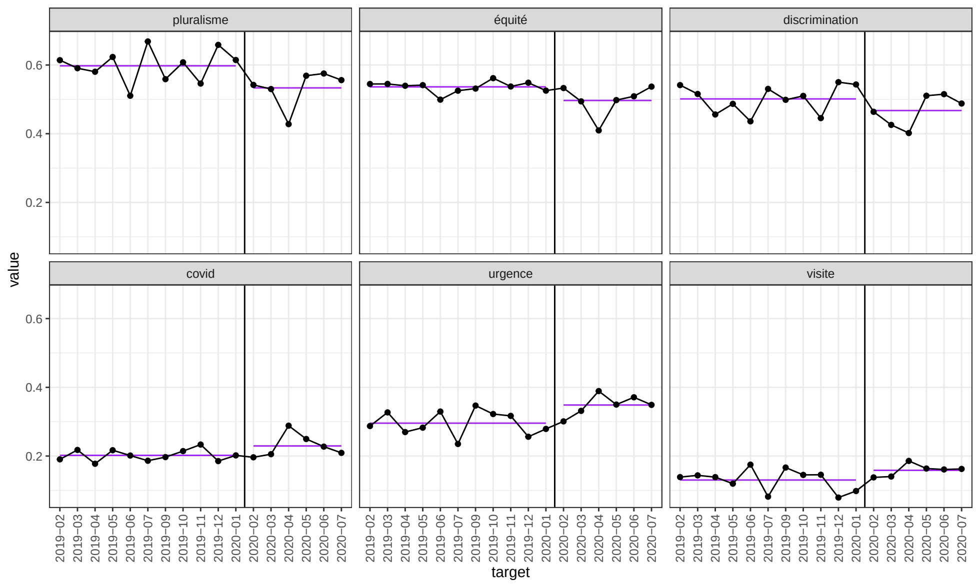


Figure 16: Average cosine similarity between ALC embeddings of `liberté` and pre-trained embeddings of relevant terms by month.

The second example uses embedding regressions to illustrate how the 2015 refugee crisis in Europe altered partisan differences in debates around immigration issues in Italy's federal parliament. Using text from all parliamentary speeches for 2014-2017 ($N = 20,747$), we regress the ALC embeddings for immigration-related terms (i.e., `immigrati, immigrazione, immigrato, immigrate, immigrazioni`) across 6-month periods on a binary indicator for whether the speaker's party is part of the government or opposition. The multivariate regression analogy is

$$\mathbf{Y} = \beta_0 + \beta_1 Government + \mathbf{E} \tag{3}$$

Figure (17) depicts the norm of β_1 for each period. When the estimate increases, this indicates that the use of `immigr*` becomes less similar across government and opposition parties. The estimates show that speakers from different parliamentary camps differ throughout the entire period, and most strongly in the months between September and December 2015—a period with large and unexpected waves of refugees arriving in Southern Europe. Figure (18) further highlights that this discontinuity in semantic differences is indeed meaningful. The figure shows terms that are most closely related to opposition and government parties in relation to immigration issues before and after the large influx of refugees. Specifically, we show the cosine similarity ratio of the ALC embeddings for immigration-related terms across opposition and government parties shortly before (subfigure (a)) and after (subfigure (b)) the refugee crisis began. In early 2015, both types of parliamentary camps discussed issues of immigration in similar ways, often sharing nearest neighbors such as emergency (`emergenziale`) or applicants (`richiedenti`). In the later months of 2015, in contrast, the vocabularies radically differ between government and opposition parties. While opposition parties still seem to talk about immigration in more general terms (e.g., invoking terms lexically related to `immigrazione`), government parties now mention normative challenges of immigration as well as legal constraints, e.g., the Schengen area or the "Bossi-Fini law". It is worth noting that we excluded stop words from the Italian parliamentary corpus to improve the performance of ALC in this case. It is possible that excluding stop words can "help" the transformation matrix in screening out common directions in the embedding space, and users may want to test the importance of removing vs. keeping stop words in their relevant language and use case. Taken together, these two examples across different parliamentary settings highlight the power of ALC to capture and illustrate semantic differences across time and groups.

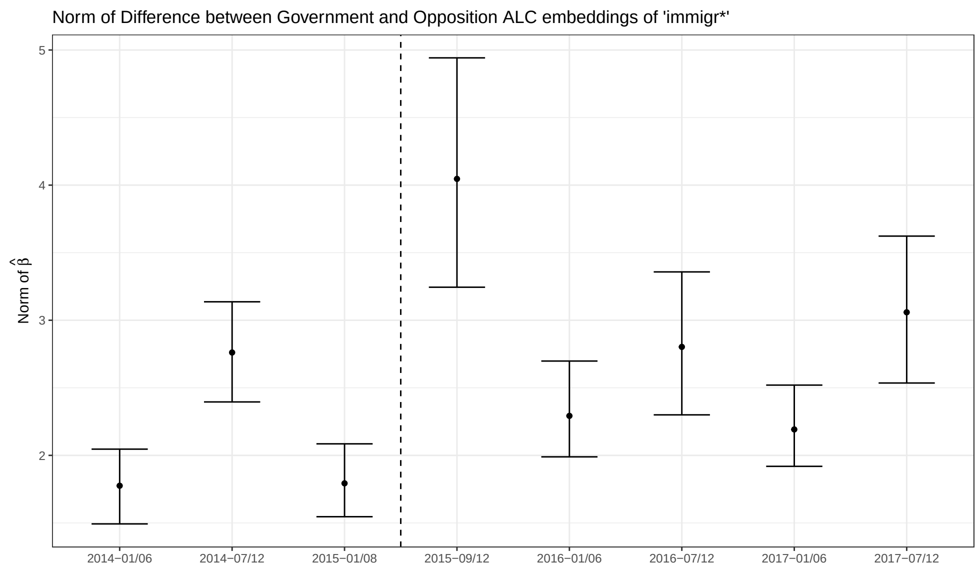
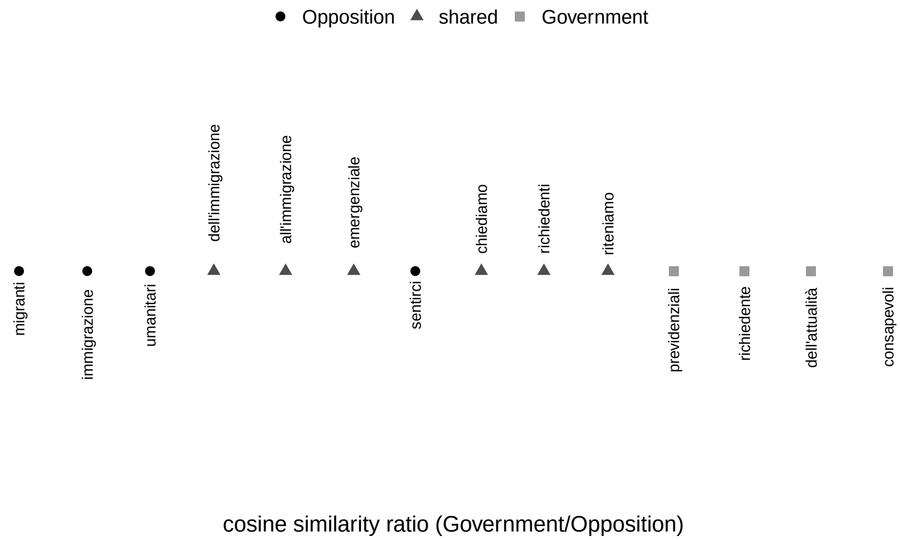
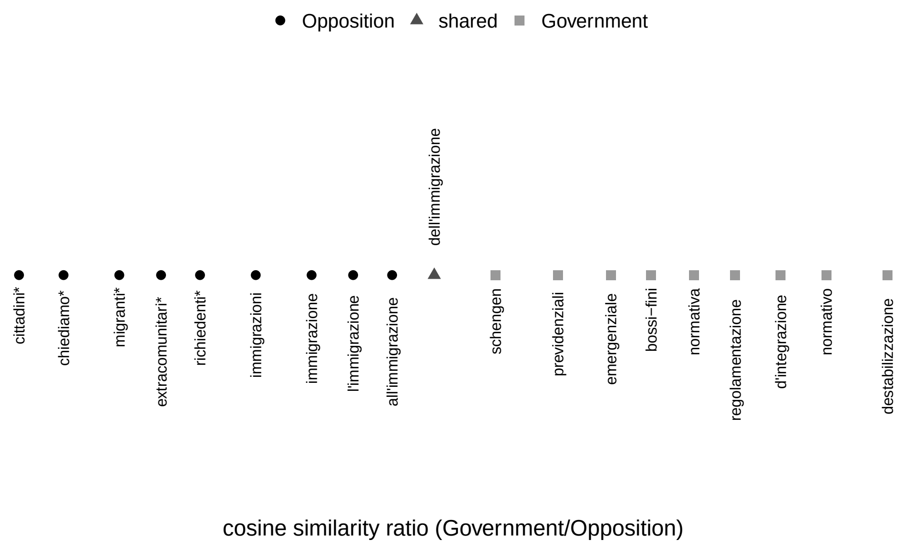


Figure 17: Relative semantic shift of `immigr*` between government and opposition parties.



(a) 2015-01/08



(b) 2015-09/12

Figure 18: Discussion of immigration diverged between government and opposition parties after the 2015 European refugee crisis

Readers may reasonably ask whether fitting the **A** matrix locally in this case would have resulted in "better" (more locally precise) embeddings. Our answer here is "no", as Table (8) shows. The table compares our pre-trained quantities and their application with locally trained embeddings to the French parliamentary debates. Columns 1 and 3 list the nearest neighbors for `liberté` for the pre-trained embeddings (our `fastText` and locally trained `GloVe`), and columns 2 and 4 show the nearest neighbors for the corresponding ALC embeddings of `liberté`. Evidently, our fastText resources capture meaningful connotations of the keyword, both for pre-trained and ALC embeddings. In contrast, locally trained quantities work well for GloVe but not for its ALC version. That is, inspecting column 4, we see that the nearest neighbors for the ALC embedding of liberté depict only function words, such as encore or aussi. Note that we excluded stop words in the underlying parliamentary corpus (except for the training of the GloVe model) to facilitate a better local fit. So, while our general suggestion is to fit the relevant quantities locally if the corpus is large enough, in this particular case, that size requirement was not fulfilled.

our fT	our fT-ALC	local GloVe	local GloVe-ALC
liberté	l'irresponsabilité	liberté	c'est
libertés	non-discrimination	d'expression	aussi
d'expression	l'impartialité	droit	tout
démocratie	d'impartialité	respect	aujourd'hui
conditionelle	pluralisme	principe	car
légalité	légalité	contraire	surtout
pluralisme	d'exigence	toute	bien
laïcité	d'autrui	garantir	fait
dignité	contrevient	leur	encore
l'égalité	l'inconstitutionnalité	choisier	faire

Table 8: Nearest neighbors for liberté for different pretrained embeddings and transformation matrices. The ALC embeddings, and the local GloVe model, use the French parliamentary corpus from Erjavec et al. (2023), 2017-2020.

To illustrate this later point, we juxtapose our results in Table 8 with a parallel exercise with the ALC embeddings for the term freedom using the Congressional records (Session 111-114) (Gentzkow, Shapiro and Taddy, 2018) in Table 9. Evidently, all sets of embeddings, i.e., the fastText pre-trained embeddings, the locally trained GloVe embeddings, and their respective ALC embeddings, return meaningful and very similar nearest neighbors for freedom. This implies that the locally fit quantities do not lag behind the pre-trained resources, provided the local corpus provides sufficient data to estimate high-quality embeddings and transformation matrices.

our fT	our fT-ALC	local GloVe	local GloVe-ALC
freedom	freedom	freedom	freedom
freedoms	conscience	liberty	liberty
liberty	freedoms	free	rights
liberties	civility	rights	religious
equality	liberties	freedoms	freedoms
conscience	democracy	right	free
democracy	humanitarianism	nation	democracy
independence	compassionately	world	fundamental
rights	equality	american	principles
autonomy	uscirf	democracy	expression

Table 9: Nearest neighbors for `freedom` for different pretrained embeddings and transformation matrices. The ALC embeddings and the local `GloVe` model use the Congressional corpus from Gentzkow, Shapiro and Taddy (2018).

References

Chang, Pi-Chuan, Michel Galley and Christopher D. Manning. 2008. Optimizing Chinese Word Segmentation for Machine Translation Performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*. StatMT '08 USA: Association for Computational Linguistics p. 224–232.

Erjavec, Tomaž, Maciej Ogrodniczuk, Petya Osenova, Nikola Ljubešić, Kiril Simov, Andrej Pančur, Michał Rudolf, Matyáš Kopp, Starkadur Barkarson, Steinþór Steingrímsson, Çağrı Çöltekin, Jesse de Does, Katrien Depuydt, Tommaso Agnoloni, Giulia Venturi, María Calzada Pérez, Luciana D. de Macedo, Costanza Navarretta, Giancarlo Luxardo, Matthew Coole, Paul Rayson, Vaidas Morkevičius, Tomas Krilavičius, Roberts Darģis, Orsolya Ring, Ruben van Heusden, Maarten Marx and Darja Fišer. 2023. "The ParlaMint corpora of parliamentary proceedings." *Language Resources and Evaluation* 57(1):415–448.
URL: *https://doi.org/10.1007/s10579-021-09574-0*

Gentzkow, Matthew, Jesse M. Shapiro and Matt Taddy. 2018. Congressional Record for the 43rd-114th Congresses: Parsed Speeches and Phrase Counts. https://data.stanford.edu/congress_text.

Grave, Edouard, Piotr Bojanowski, Prakhar Gupta, Armand Joulin and Tomás Mikolov. 2018. "Learning Word Vectors for 157 Languages." *CoRR* abs/1802.06893.

Høyland, Bjørn, Indraneel Sircar and Simon Hix. 2009. "Forum section: an automated database of the european parliament." *European Union Politics* 10(1):143–152.

Khodak, Mikhail, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart and Sanjeev Arora. 2018. "A La Carte Embedding: Cheap but Effective Induction of Semantic Feature Vectors." *CoRR* abs/1805.05388.

Kudo, Takumitsu. 2005. MeCab : Yet Another Part-of-Speech and Morphological Analyzer.

Pennington, Jeffrey, Richard Socher and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pp. 1532–1543.

Rodriguez, Pedro L and Arthur Spirling. 2022. "Word embeddings: What works, what doesn't, and how to tell the difference for applied research." *The Journal of Politics* 84(1):101–115.

Rodriguez, Pedro L, Arthur Spirling and Brandon M Stewart. 2023. "Embedding Regression: Models for Context-Specific Description and Inference." *American Political Science Review* pp. 1–20.

Rui, Ming. 2020. ICU tokenizer.