Supplementary Materials
Cross-lingual classification of political texts using multilingual sentence embeddings

Hauke Licht[*]

May 11, 2022

Contents

[*]University of Cologne, Cologne Center for Comparative Politics. Contact: hauke.licht@wiso.uni-koeln.de

A Input alignment through machine translation: a running example

The fundamental challenge in cross-lingual quantitative text analysis is language-independent inference of latent concepts from multilingual corpora. The two sentences listed in Example S.1 illustrate this challenge: Both are statements about unemployment, and in both, the author is pledging to lower unemployment. Consequently, as shown in Example S.2, the bag-of-words representations of these sentences exhibit no vocabulary overlap.

Example S.1. Sentences in multilingual example corpus.

	Language	Text
`doc`$_1$	English	'We will fight unemployment.'
`doc`$_2$	German	'Wir werden die Arbeitslosigkeit reduzieren.'

Example S.2. Bag-of-words representations of sentences in Example S.1 after text pre-processing (lowercasing and punctation removal).

	`we`	`will`	`fight`	`unemployment`	`wir`	`werden`	`die`	`arbeitslosigkeit`	`reduzieren`
`doc`$_1$	1	1	1	1	0	0	0	0	0
`doc`$_2$	0	0	0	0	1	1	1	1	1

Note: All tokens have been lowercased.

Full-text machine translation provides a remedy to this problem. This is shown Example S.3. When full-text translated documents are pre-processed and tokenized into n-gram tokens, this results in monolingual bag-of-words representations of originally multilingual documents (cf. Example S.4). Compared to the bag-of-words representations of untranslated sentences (Example S.2), these representations encode the semantic similarity between our two example sentences.

Example S.3. Sentences in Example S.1 after translating non-English texts into English.

ID	Original text	Transfer	Text (in English)
`doc`$_1$	'We will fight unemployment.'	$\xrightarrow{\text{as-is}}$	'We will fight unemployment.'
`doc`$_2$	'Wir werden die Arbeitslosigkeit reduzieren.'	$\xrightarrow{\text{translate}}$	'We will reduce unemployment.'

Example S.4. Bag-of-words representations of sentences Example S.1 after applying full-text translation approach.

	`we`	`will`	`fight`	`unemployment`	`reduce`
$\texttt{doc}_1$	1	1	1	1	0
$\texttt{doc}_2'$	1	1	0	1	1

Note: All tokens have been lowercased.

With *token translation*, texts are first tokenized in their original language, and only the tokens in the resulting language-specific bag-of-words are translated. This procedure is illustrated in Example S.5. As shown in Example S.6, token translation enables representing documents as bag-of-words in the target language. And again, this representation encodes well the semantic similarity between our two example sentences.

However, translating words or phrases outside the textual contexts in which they are used can result in translations errors. For instance, such a translation error occurs in Example S.5 in case of the German word 'werden': Read in context, it should translate to 'will,' not 'become.' Comparing Examples S.3 and S.6 shows that this error would not have been committed with full-text translation.

Example S.5. Applying token translation approach to non-English sentence in Example S.1. For source tokens, see columns 5–9 of Example S.2.

	$\texttt{doc}_2$: *'Wir werden die Abeitslosigkeit reduzieren.'*				
Source token:	'wir'	'werden'	'die'	'arbeitslosigkeit'	'reduzieren'
English translation:	'we'	'become'	'the'	'unemployment'	'reduce'

Note: All tokens have been lowercased.

Example S.6. Bag-of-words representations of sentences in Example S.1 after applying token-translation approach.

	`we`	`will`	`fight`	`unemployment`	`become`	`the`	`reduce`
$\texttt{doc}_1$	1	1	1	1	0	0	0
$\texttt{doc}_2''$	1	0	0	1	1	1	1

Note: All tokens have been lowercased.

B Multilingual sentence embedding models

In section 2.2 of the main paper, I have provided a high-level introduction to the intuition behind multilingual sentence embedding (MSE). In section 3.3, I have briefly described the pre-trained multilingual embedding models used in my study. Below, I describe in greater detail the neural network architectures and training strategies used to obtain these models and provide Python code examples that show how these models can be used for off-the-shelf MSE.

B.1 The Language-Agnostic Sentence Embedding Representation model

The *Language-Agnostic Sentence Embedding Representations* (LASER) model proposed by Artetxe and Schwenk (2019) is an encoder–decoder architecture for learning multilingual sentence representations. Provided with parallel sentences, that is, versions of otherwise identical texts in two different languages (also called *bitexts*), this model is trained for sequence-to-sequence modeling. Put simply, it learns to translate (cf. Schwenk and Douze, 2017). A visual depiction of the LASER model architecture is shown in Figure S.1.

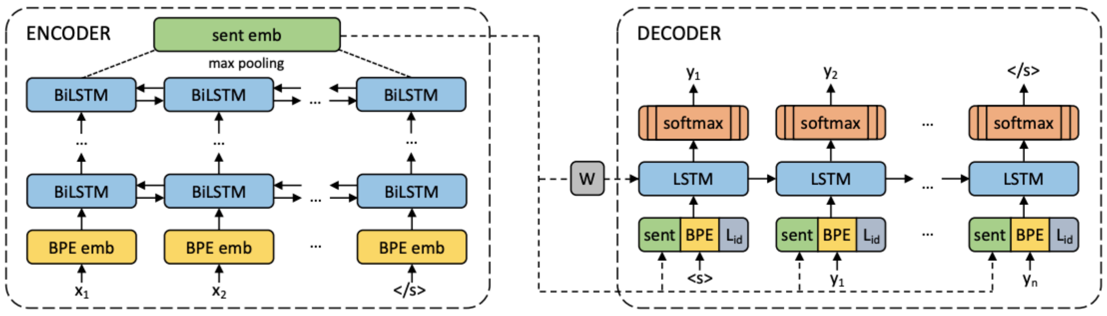

Figure S.1. LASER model architecture (Artetxe and Schwenk, 2019, Figure 1)

The task of the LASER encoder is to generate a vector representation ("encoding") for a sentence in its source language ($\mathbf{x}_{1:n}$). The decoder is tasked to reconstruct the sentence in its other language version—the output or target sequence ($\mathbf{y}_{1:n}$)—as accurately as possible.

Predicting tokens in the output sequence allows computing the cross-entropy loss, which is used to jointly optimize the encoder and decoder. Once training is completed, the decoder is discarded, and the encoder can be used for MSE.

Artetxe and Schwenk (ibid.) pre-train their LASER model on 6 large parallel corpora (cf. their Appendix A). This results in the language support reported in Table S.1.

Table S.1. Language support of the pre-trained LASER model provided by Vaginay (2020). Columns 2–3 report ISO 369 language codes. Source: https://github.com/facebookresearch/LASER

Name			*Name*			*Name*		
Afrikaans	`afr`	`af`	French	`fre`	`fr`	Marathi	`mar`	`mr`
Albanian	`alb`	`sq`	Galician	`glg`	`gl`	Norwegian (Bokmål)	`nob`	`nb`
Amharic	`amh`	`am`	Georgian	`geo`	`ka`	Occitan	`oci`	`oc`
Arabic	`ara`	`ar`	German	`ger`	`de`	Persian (Farsi)	`peo`	
Armenian	`arm`	`hy`	Greek	`ell`	`el`	Polish	`pol`	`pl`
Aymara	`aym`	`ay`	Hausa	`hau`	`ha`	Portuguese	`por`	`pt`
Azerbaijani	`aze`	`az`	Hebrew	`heb`	`he`	Romanian	`ron`	`ro`
Basque	`baq`	`eu`	Hindi	`hin`	`hi`	Russian	`rus`	`ru`
Belarusian	`bel`	`be`	Hungarian	`hun`	`hu`	Serbian	`srp`	`sr`
Bengali	`ben`	`bn`	Icelandic	`ice`	`is`	Sindhi	`snd`	`sd`
Berber languages	`ber`		Ido	`ido`	`io`	Sinhala	`sin`	`si`
Bosnian	`bos`	`bs`	Indonesian	`ind`	`id`	Slovak	`slo`	`sk`
Breton	`bre`	`br`	Interlingua	`ina`	`ia`	Slovenian	`slv`	`sl`
Bulgarian	`bul`	`bg`	Interlingue	`ile`	`ie`	Somali	`som`	`so`
Burmese	`bur`	`my`	Irish	`gle`	`ga`	Spanish	`spa`	`es`
Catalan	`cat`	`ca`	Italian	`ita`	`it`	Swahili	`swa`	`sw`
Central/Kadazan Dusun	`dtp`		Japanese	`jpn`	`ja`	Swedish	`swe`	`sv`
Central Khmer	`khm`	`km`	Kabyle	`kab`		Tagalog	`tgl`	`tl`
Chavacano	`cbk`		Kazakh	`kaz`	`kk`	Tajik	`tgk`	`tg`
Chinese	`chi`	`zh`	Korean	`kor`	`ko`	Tamil	`tam`	`ta`
Coastal Kadazan	`kzj`		Kurdish	`kur`	`ku`	Tatar	`tat`	`tt`
Cornish	`cor`	`kw`	Latvian	`lav`	`lv`	Telugu	`tel`	`te`
Croatian	`hrv`	`hr`	Latin	`lat`	`la`	Thai	`tha`	`th`
Czech	`cze`	`cs`	Lingua Franca Nova	`lfn`		Turkish	`tur`	`tr`
Danish	`dan`	`da`	Lithuanian	`lit`	`lt`	Uighur	`uig`	`ug`
Dutch	`nld`	`nl`	Low German/Saxon	`nds`		Ukrainian	`ukr`	`uk`
Eastern Mari	`mhr`		Macedonian	`mac`	`mk`	Urdu	`urd`	`ur`
English	`eng`	`en`	Malagasy	`mlg`	`mg`	Uzbek	`uzb`	`uz`
Esperanto	`epo`	`eo`	Malay	`may`	`ms`	Vietnamese	`vie`	`vi`
Estonian	`est`	`et`	Malayalam	`mal`	`ml`	Wu Chinese	`wuu`	
Finnish	`fin`	`fi`	Maldivian (Divehi)	`div`	`dv`	Yue Chinese	`yue`	

Sentence embedding using a pre-trained LASER encoder

Sentences can be embedded using the pre-trained LASER encoder made available through the `laserembeddings`[1] Python package (Vaginay, 2020). The `laserembeddings` package makes only minor adaptations to the original implementation during text pre-processing.[2] Vaginay (ibid.) shows that these adaptations are inconsequential for sentence embedding in the languages studied in this paper.[3]

Listing B.1. Shell code for setting up the LASER model made available by the `laserembeddings` Python package.

```
# install the package
pip3 install laserembeddings > /dev/null
# download the pre-trained LASER encoder
python3 -m laserembeddings download-models
```

As shown in Listing B.1, the setup involves installing the package[4] and downloading the pre-trained model. As shown in Listing B.2, after importing the `Laser` class from the `laserembeddings` module (line 2), it can be instantiated (as `model` in line 5) to load the pre-trained LASER encoder. The `embed_sentences` method can then be used to embed texts (see line 11). Note that one also needs to indicate the language(s) of the input sentence(s). However, this information is only required for correct pre-processing; the LASER encoder itself has not received input language information during pre-training, nor does it require this information at inference/prediction time.

Listing B.2. Python code for sentence embedding using the `laserembeddings` package.

```
1  # load required modules
2  from laserembeddings import Laser
3
4  # instantiate pre-trained LASER encoder
5  model = Laser()
6
7  texts = ["Hello world!", "Hallo Welt!"]
8  langs = ["eng", "deu"]
9
10 # embed
11 embeddings = laser.embed_sentences(texts, lang = langs)
```

[1] https://github.com/yannvgn/laserembeddings
[2] see Vaginay (2020): "Differences with the original implementation"
[3] see Vaginay (ibid.): "Comparison with LASER"
[4] for details see https://github.com/yannvgn/laserembeddings#installation

B.2 Knowledge-distilled models

Reimers and Gurevych (2020) propose to extend existing (multilingual) sentence embedding models to new languages by a "knowledge distillation" procedure: a pre-trained sentence embedding model (the "teacher") teaches a MSE model (the "student") how to represent sentences in different languages. This is achieved by optimizing the alignment between the teacher's and the student's sentence embeddings. More concretely, given are (i) the "teacher" model M capable of sentence embedding in one or more source languages s and (ii) a corpus $\mathcal{C}$ of parallel sentences mapping sentences in the source language(s) to multiple target languages (t), $\{(s_i, t_i)\}_{i \in \mathcal{C}}$. A MSE model (the "student," $\hat{M}$) is then trained jointly with the teacher to maximize alignment (a) between the teacher and student models' source sentences embeddings *and* (b) between the source sentence embeddings of the teacher model and the target sentences embedding of the student models. This is achieved with stochastic gradient descent methods by minimizing the mean squared loss for parallel sentences in a mini batch $\mathcal{B}$:

$$\frac{1}{\mathcal{B}} \sum_{j \in \mathcal{B}} \left[\left(M(s_j) - \hat{M}(s_j) \right)^2 + \left(M(s_j) - \hat{M}(t_j) \right)^2 \right]$$

A visual depiction of the knowledge distillation architecture is shown in Figure S.2.

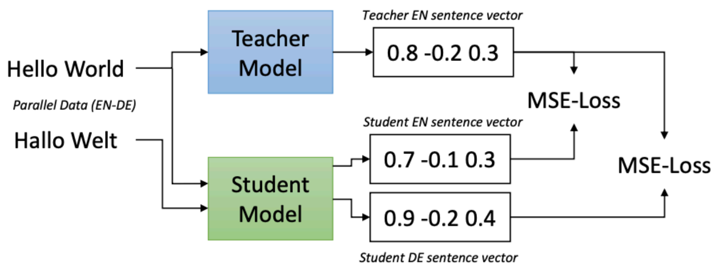

Figure S.2. Neural network architecture for learning multilingual sentence embedding models through knowledge distillation (ibid., Figure 1)

Reimers and Gurevych (ibid.) emphasize that knowledge-distilled MSE models exhibit two desirable properties: First, the embedding space is aligned across languages. Hence, the embeddings of identical sentences in different languages are close to one another. Second, the vector space properties of the teacher model are transferred to the student model and hence maintained when embedding sentences in new target languages. This second property comes in particularly handy when using a sentence embedding models pre-trained with a paraphrasing objective as teacher model: The resulting knowledge-distilled multilingual embedding models perform well in paraphrasing, unlike LASER, which is optimized for exact translation.

Pre-trained knowledge-distilled models

In their paper, Reimers and Gurevych (2020) rely on the English sentence-BERT model (Reimers and Gurevych, 2019) as teacher and use different MSE models as students.

Knowledge-distilled XLM-R Among their student models is the XLM-RoBERTa (XLM-R) model. XLM-R is a transformer-based multilingual masked language model proposed by Conneau et al. (2020) that is trained with the (self-supervised) multilingual masked language modeling objective proposed by Lample and Conneau (2019). To produce a knowledge-distilled version of XLM-R, Reimers (2021) uses a RoBERTa sentence embedding model[5] pre-trained on English paraphrasing data and parallel data covering the languages and corpora reported in Table S.2. Reimers (ibid.) makes this model publicly available for download via the `sentence-transformers` Python package.[6]

Knowledge-distilled mUSE In addition, Reimers (ibid.) also makes available a sentence embedding model that was distilled using the multilingual Universal Sentence Encoder (mUSE, Yang et al., 2020) as a teacher.[7] mUSE is designed for obtaining "universal" sentence embeddings (i.e., embeddings that enable good performance in a wide range of tasks Cer et al., 2018) and relies on multi-task training of a dual encoder architecture to achieve this (Chidambaram et al., 2019). Yang et al. (2020) train mUSE to bitexts covering 16 languages (ibid., Table 1) and show that the MSEs it obtains indeed make for very good features in a wide range of natural language understanding tasks. Reimers (2021) extends this model to new languages using the parallel corpora reported in Table S.2.

Applying knowledge distillation to the original mUSE model using 8 parallel corpora and English as source language (see Table S.2), Reimers (2021) has created a knowledge-distilled mUSE model that covers the 51 languages reported in Table S.3 and is publicly available for download via the `sentence-transformers` Python package (see Listing B.3).

Sentence embedding using knowledge-distilled models

Listing B.4 shows how multiple sentences can be embedded using the knowledge-distilled mUSE model. First, one needs to import the `SentenceTransformer` class (line 2 of Listing B.4) and call it to create an instance of the knowledge-distilled mUSE model, here named `model` (line 5). This will automatically download the pre-trained model. Calling `SentenceTransformer` with `"paraphrase-xlm-r-multilingual-v1"` instead will load

[5] `paraphrase-distilroberta-base-v1`
[6] `paraphrase-xlm-r-multilingual-v1`
[7] `distiluse-base-multilingual-cased-v2`

Table S.2. Parallel corpora and target languages used to train the knowledge-distilled models. Please refer to Reimers and Gurevych (2020) for references of these corpora.

Corpus	Target languages
Europarl (en – .)	bg, cs, da, de, el, es, et, fi, fr, hu, it, lt, lv, nl, pl, pt, ro, sk, sl, sv
GlobalVoices (en – .)	ar, bg, ca, cs, da, de, el, es, fa, fr, he, hi, hu, id, it, ko, mk, my, nl, pl, pt, ro, ru, sq, sr, sv, tr, ur
JW300 (en – .)	ar, bg, cs, da, de, el, es, et, fa, fi, fr, gu, he, hi, hr, hu, hy, id, it, ja, ka, ko, lt, lv, mk, mn, mr, my, nl, pl, pt, ro, ru, sk, sl, sq, sv, th, tr, uk, ur, vi
News-Commentary (en – .)	ar, cs, de, es, fr, it, ja, nl, pt, ru
OpenSubtitles (en – .)	ar, bg, ca, cs, da, de, el, es, et, fa, fi, fr, gl, he, hi, hr, hu, hy, id, it, ja, ka, ko, lt, lv, mk, ms, nl, pl, pt, ro, ru, sk, sl, sq, sr, sv, th, tr, uk, ur, vi, zh_cn
Tatoeba (en – .)	ara, bul, cat, ces, cmn, dan, deu, ell, est, fin, fra, glg, guj, heb, hin, hrv, hun, hye, ind, ita, jpn, kat, kor, kur, lit, lvs, mar, mkd, mon, mya, nld, nob, pes, pol, por, ron, rus, slk, slv, spa, sqi, srp, swe, tha, tur, ukr, urd, vie, zsm
TED2020 (en – .)	ar, bg, ca, cs, da, de, el, es, et, fa, fi, fr-ca, fr, gl, gu, he, hi, hr, hu, hy, id, it, ja, ka, ko, ku, lt, lv, mk, mn, mr, ms, my, nb, nl, pl, pt-br, pt, ro, ru, sk, sl, sq, sr, sv, th, tr, uk, ur, vi, zh-cn, zh-tw
WikiMatrix (en – .)	ar, bg, ca, cs, da, de, el, es, et, fa, fi, fr, gl, he, hi, hr, hu, id, it, ja, ka, ko, lt, mk, mr, nl, pl, pt, ro, ru, sk, sl, sq, sr, sv, tr, uk, vi, zh

Table S.3. Language support of the knowledge-distilled models (Reimers, 2021).

Language	Codes	Language	Codes	Language	Codes
Albanian	sq, sqi	Georgian	ka, kat	Mongolian	mn, mon
Arabic	ar, ara	German	de, deu	Norwegian Bokmål	nb, nob
Armenian	hy, hye	Greek	el, ell	Persian	fa, pes
Bulgarian	bg, bul	Gujarati	gu, guj	Polish	pl, pol
Burmese	my, mya	Hebrew	he, heb	Portuguese	por, pt, pt-br
Catalan	ca, cat	Hindi	hi, hin	Romanian	ro, ron
Chinese (Cantonese)	zh, zh_cn, zh-cn	Hungarian	hu, hun	Russian	ru, rus
Chinese (Mandarin)	cmn	Indonesian	id, ind	Serbian	sr, srp
Chinese (Taiwanese)	zh-tw	Italian	it, ita	Slovak	sk, slk
Croatian	hr, hrv	Japanese	ja, jpn	Slovenian	sl, slv
Czech	cs, ces	Korean	ko, kor	Spanish	es, spa
Dansk	da, dan	Kurdish	ku, kur	Swedish	sv, swe
Dutch	nl, nld	Latvian	lv, lvs	Thai	th, tha
Estonian	est, et	Lithuanian	lit, lt	Turkish	tr, tur
Finnish	fi, fin	Macedonian	mk, mkd	Ukrainian	uk, ukr
French	fr, fr-ca, fra	Malay	ms, zsm	Urdu	ur, urd
Galician	gl, glg	Marathi	mr, mar	Vietnamese	vi, vie

and instantiate the knowledge-distilled XLM-R model. Finally, the `encode` method of the `SentenceTransformer` instance can be used to embed texts (line 10).

Listing B.3. Shell code to install the `sentence-transformers` Python package.

```
1 # install (in shell/bash)
2 pip3 install sentence-transformers==0.4.1 > /dev/null # capture output
```

Listing B.4. Sentence embedding using the `sentence-transformers` Python package.

```
 1 # import required module
 2 from sentence_transformers import SentenceTransformer
 3 
 4 # instantiate embedding model
 5 model = SentenceTransformer("distiluse-base-multilingual-cased-v2")
 6 
 7 texts = ["Hello world!", "Hallo Welt!"]
 8 
 9 # embed (optionally: to return a numpy array, set convert_to_numpy = True)
10 embeddings = model.encode(texts)
```

B.3 Practical considerations

How long does it take to embed sentences with these models?

To get an idea of how fast MSE is compared to machine translation, I have sampled 750 sentences per language from the Düpont and Rachuj (2022) datasets, embedded and translated them, and recorded the time elapsed for embedding/translating 750 sentences. Note that I have run this experiment using a GPU on *Google Colab*.

For LASER, the median time elapsed while embedding 750 sentences is only 2.5 seconds and the first and third quartile values are 2 and 3, respectively. For the mUSE and XLM-R, these values are 4.5, 3.75, and 5.

In contrast, the median time elapsed while translating 750 sentences with the open-source M2M machine translation model is 416 seconds and the first and third quartile values are 289.25 and 477, respectively. Thus, one can embed 92.44 sentences with the mUSE or XLM-R models for every sentence translated with M2M.

How long is "sentence-like?"

The neural network architectures underpinning the MSE models discussed above have a maximum number of tokens they can ingest: The implementation of LASER by Vaginay (2020). And the knowledge-distilled models provided by Reimers (2021) 128 tokens.

One token does not correspond to a word, however, because the tokenizers of these models are "subword" tokenizers trained on the corpora used for pre-training. To get an idea of how many words a text can count to allow embedding without truncation, I have again taken the sample of 750 sentences per language stemming from the Düpont and Rachuj (2022) datasets, tokenized them with the given models' tokenizers, and computed the token-to-word ratio for each. Note that I have run this experiment using a GPU on Google Colab.

Figure S.3 reports these numbers. Note that I only report numbers for LASER and mUSE, because mUSE and XLM-R models have been trained on the same data and hence their tokenizers behave identically.

For LASER's tokenizer, the median token-to-word ration is 1.67 and the first and third quartile values are 1.47 and 1.95, respectively. For the tokenizer of mUSE and XLM-R, these values are 1.5, 1.36, and 1.74.

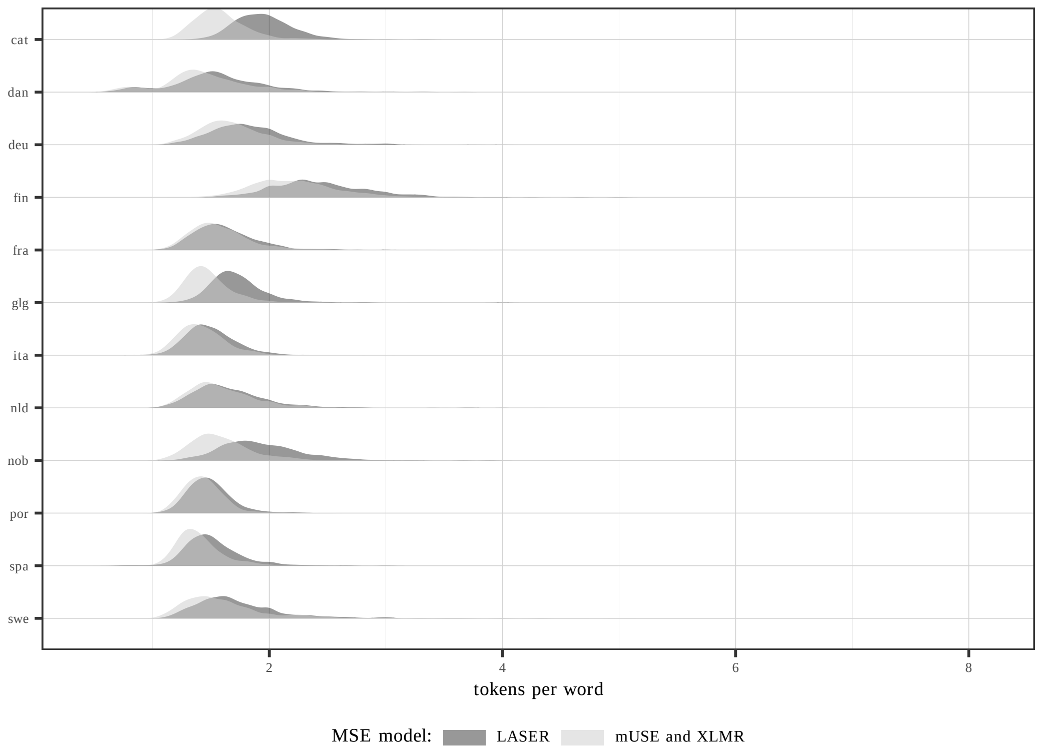

Figure S.3. Token-to-words ratio when tokenizing sentences sampled from the Düpont and Rachuj (ibid.) datasetswith pre-trained multlingual sentence embedding (MSE) models.

C Datasets

In my analyses I combine two data sources with primary data recorded in the *Comparative Manifestos Project* (CMP) corpus (Volkens et al., 2020). First, a dataset compiled by Düpont and Rachuj (2022) that provides full-text translations of the sentences of a subset of manifestos in the CMP corpus that covers twelve different European languages. Second, a dataset compiled by Lehmann and Zobel (2018) that records immigration/integration issue codings obtained for a sample of manifesto quasi-sentences in the original CMP corpus that covers eight different European languages. I describe these data sources below.

C.1 The Comparative Manifestos Project corpus

The Comparative Manifestos Project (CMP) corpus (Volkens et al., 2020) is a collection of election manifestos of political parties from developed and developing democracies that have been annotated by country experts in their original languages. Version 2020a of the CMP corpus comprises a total of 2576 party manifestos, 1427 of which are annotated at the quasi-sentence level. Quasi-sentences are sentence sub-units '[containing] exactly one statement or "message"' that map one-to-one or many-to-one to sentences (Werner et al., 2015, p. 6).

Data cleaning

I have not pre-processed the text of quasi-sentences in any way. The only change I have made to the data is that I have harmonized quasi-sentence level codings to comply with the fourth version of the coding scheme.[8] Harmonization of CMP codes was required because the CMP coding scheme has been repeatedly updated. I have selected the fourth version as the target scheme because it is backward compatible.

After harmonization, I have removed all quasi-sentences for which (a) the harmonized CMP code was 000 ("uncoded"), missing (`NA`), or indicated heading text ("H"); or (b) the quasi-sentence text was missing (`NA`), an empty string, or consisted only of space characters. This was necessary because codings to which exclusion criterion (a) applies are invalid, and computing feature representations of texts to which exclusion criterion (b) applies are impossible.

[8] I have relied on functions provided in the `manifestoR` package to do so (Lewandowski et al., 2020): I have first called `recode_cee_codes` on version-dependent CMP codes and then `recode_v5_to_v4` on the result of this call.

Table S.4. Mapping of CMP codes to left–right–neutral recategorization by topic categories. Note that CMP codes have been harmonized to comfort with the fourth version of the CMP coding scheme.

Position	Topic	CMP codes (harmonized)
`left`	External Relations (`extrel`)	104
	Freedom and Democracy (`freedem`)	201, 203
	Political System (`polsys`)	305
	Economy (`econ`)	401, 402, 407, 414
	Welfare and Quality of Life (`welqual`)	505
	Fabric of Society (`fabsoc`)	601, 603, 605, 606
`right`	External Relations (`extrel`)	103, 105, 106, 107
	Freedom and Democracy (`freedem`)	202
	Economy (`econ`)	403, 404, 406, 412, 413
	Welfare and Quality of Life (`welqual`)	504, 506
	Social Groups (`socgrp`)	701
`neutral`	External Relations (`extrel`)	101, 102, 108, 109, 110
	Freedom and Democracy (`freedem`)	204
	Political System (`polsys`)	301, 302, 303, 304
	Economy (`econ`)	405, 408, 409, 410, 411, 415, 416
	Welfare and Quality of Life (`welqual`)	501, 502, 503, 507
	Fabric of Society (`fabsoc`)	602, 604, 607, 608
	Social Groups (`socgrp`)	702, 703, 704, 705, 706
`uncoded`	uncoded (`uncoded`)	000

Outcome variables

I have conducted the first two analyses presented in the main paper with these data focusing on two outcome variables: (i) a 7-category topic indicator and (ii) a 3-category position indicator. Table S.4 reports the mapping of outcome labels to harmonized CMP codes. Note that I have always included the "uncoded" category as a valid label during classifier training.

C.2 Manifesto full-text translations provided by Düpont and Rachuj (2022)

To compare supervised learning from multilingual sentence embeddings to supervised learning from bag-of-words representations of machine-translated full texts in the first two analyses of the main paper, I rely on manifesto full-text translations obtained by Nils Düpont and Marting Rachuj (ibid.). Düpont and Rachuj (ibid.) study policy diffusion in a cross-lingual setting. They propose operationalizing their quantity of interest as the degree of textual similarity between any given pair party manifestos. To enable cross-lingual measurement, they compare manifestos' bag-of-words representations in a common target language—English in their case. To validate this approach, they compare measurements of textual similarity obtained from full-text translations to that of their measurement strategy.

Düpont and Rachuj (ibid.) have sampled several manifestos from 19 different coun-

Table S.5. Manifesto with and without annotations in Düpont and Rachuj's replication data by language.

	Annotated	
Language	yes	no
Catalan	1	1
Danish	10	25
Dutch	14	25
Finnish	4	12
French	8	22
Galician	2	0
German	18	26
Italian	4	16
Norwegian	5	14
Portuguese	5	8
Spanish	12	4
Swedish	5	19

tries,[9] obtained the full texts of these manifestos, segmented full texts into sentences, and submitted the resulting sentence-level data to the Google Cloud Translation API for machine translation into English. It are these sentence-level full-text translations I use in my study.

While their data is ideal for validating cross-lingual quantitative text analysis methods, Düpont and Rachuj's data collection strategy results in limitations regarding the usability of their data for text classification purposes. These limitations necessitate the following four corpus subsetting and text cleaning steps.

Discarding sentences stemming from unannotated manifestos First, Düpont and Rachuj have sampled from *all* manifestos for which full texts were available in their country-specific subset of the CMP corpus (2018a version). As a result, there are annotations available only for a subset of the manifestos Düpont and Rachuj have sampled (see Table S.5). The absence of annotations means that there are no labels to train a supervised text classifier. Hence, I had to discarded these unannotated manifestos from my analyses.

Mapping quasi-sentences to sentences Second, Düpont and Rachuj build their dataset from full texts and used automated sentence segmentation to arrive at a sentence-level corpus. However, annotations in the CMP corpus are available only at the level of *quasi-sentences.* Generally speaking, quasi-sentences map many-to-one to sentences. For any given sentence, this implies that for any outcome scheme of interest (e.g., the 7-category topic scheme), the set of codings at the *sentences* level may comprise more than

[9] Australia, Canada, Ireland, New Zealand, United Kingdom, United States, Denmark, Finland, Norway, Sweden, Austria, Belgium, France, Germany, Italy, Netherlands, Switzerland, Portugal, and Spain.

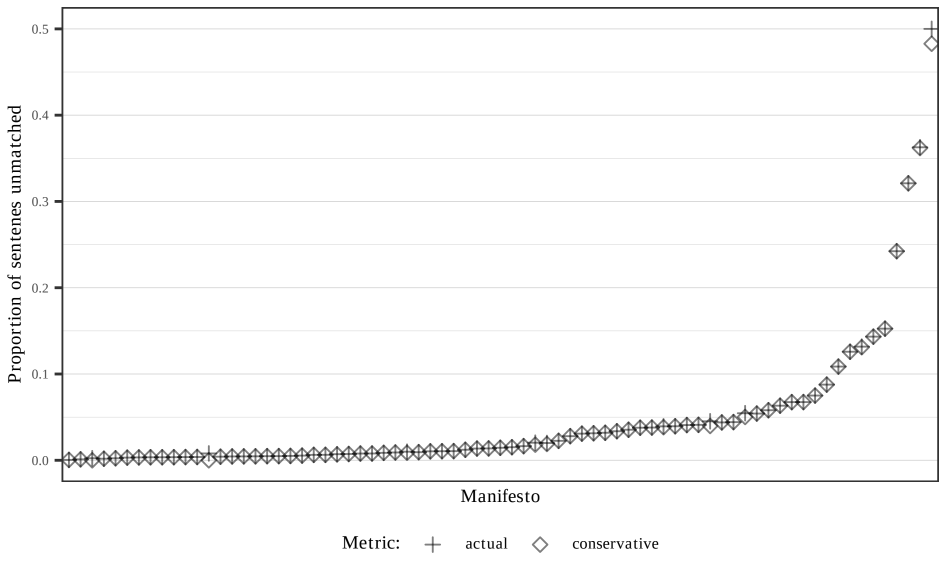

Figure S.4. Proportions of sentences in annotated manifestos that could not be matched to any quasi-sentence.Point shaped distinguish between the "actual" proportions (unmatched over all sentences)and a "conservative" metric that only looks at the proportion of unmatched sentences between the first and last matched sentence in a manifesto (e.g., to exclude often unannotated preamble text).

Table S.6. Distribution of sentences in terms of their numbers of matched of quasi-sentencens (N_{QSs}).

N_{QSs}	N	Prop.
1	63942	0.83
2	8708	0.11
3	2010	0.03
4	915	0.01
5	472	0.01
6	274	0.00
7	174	0.00
8	136	0.00
9	96	0.00
10 or more	341	0.00

one unique label. And because a sentence may comprise more than one quasi-sentences, its quasi-sentence level labels may disagree with one another.

To use Düpont and Rachuj's data despite this issue, I have matched quasi-sentences to texts represented in Düpont and Rachuj's corpus. I have developed a text-matching algorithm to this end that iterates over quasi-sentences in the CMP corpus and tries to match them to sentences in Düpont and Rachuj corpus of annotated manifestos. Applied to all annotated manifestos, this procedure resulted in large proportions of sentences–quasi-sentence matches in many manifestos. In fact, Figure S.4 shows that in only 9 manifestos 10 percent or more could not been matched. I have discarded these manifestos from my analyses.

Table S.7. Distribution of unambiguous topic and position codings (column panels) in quasi-sentences matched to sentences by numbers of matched quasi-sentences (N_{QSs}).

	Unambiguous "topic" codings						Unambiguous "position" codings				
N_{QSs}	0	1	2	3	4	5	0	1	2	3	4
1	934	63008	0	0	0	0	1017	62925	0	0	0
2	70	7294	1344	0	0	0	79	7375	1254	0	0
3	10	1324	572	104	0	0	16	1385	555	54	0
4	7	520	270	105	13	0	8	540	299	68	0
5	1	211	160	88	11	1	1	231	191	48	1
6	2	128	93	36	14	1	2	141	100	31	0

Identifying unambiguously coded sentences When looking at the distribution of numbers of matched quasi-sentences per sentence across sentences with at least one match (Table S.6), we see that the overwhelming number of sentences (96.9%) are matched to only one, two or three sentences. This is good news given that for these sentences identifying an unambiguous topic or position coding should prove not too difficult. In fact, as Table S.7 shows, in the set of sentences matched to up to six quasi-sentences (the 99% percentile), there are 72485 sentences with unambiguous topic (CMP domain) codings and are 72597 sentences with unambiguous position codings. These are the sentences I can base my comparative classification experiments on because any disambiguation rule applied to sentences matched to quasi-sentences with more than one coding at the sentence level would be arbitrary and introduce (additional) noise in labels.

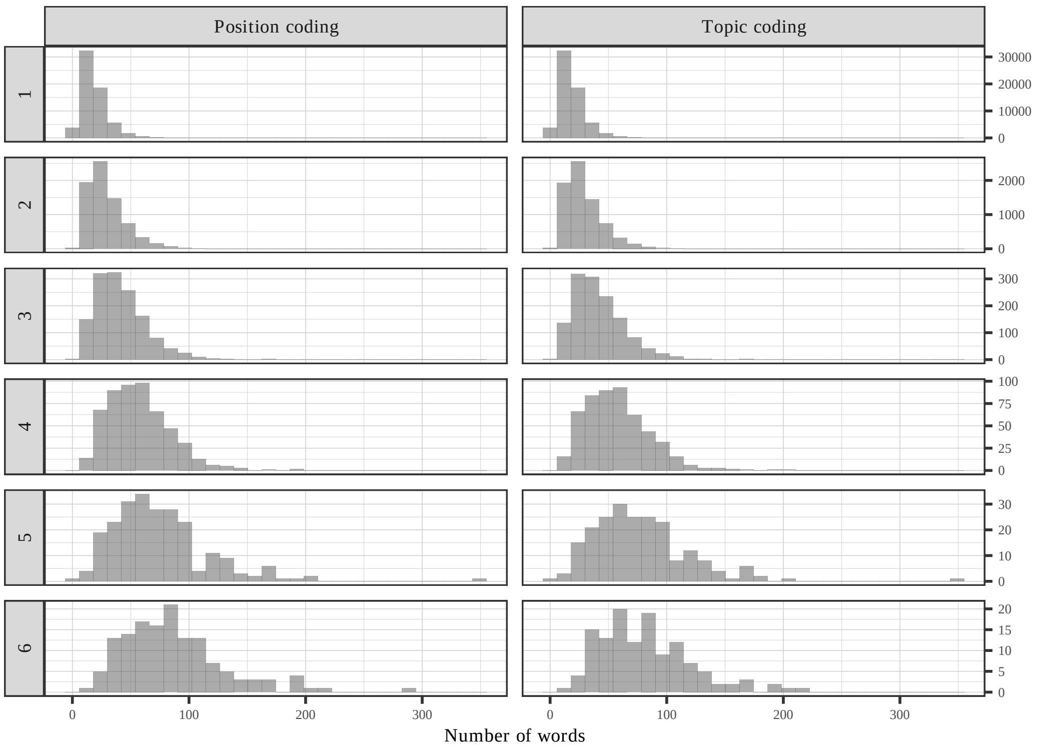

Figure S.5. Distribution of numbers of words in sentences by numbers of matched quasi-sentences and outcome variable.

Table S.8. Cross-tabulation of sentences by whether or not they are unambiguously position and/or topic coded.

	Unambiguously topic coded	
Unambiguously position coded	no	yes
no	4723	630
yes	716	70999

Table S.9. Distribution of sentence level outcome lables across languages in data retained from (Düpont and Rachuj, 2022).

	Topic categories							*Position categories*			
	extrel	freedem	polsys	econ	welqual	fabsoc	socgrp	left	neutral	right	uncoded
Catalan	65	244	134	465	642	96	118	135	736	893	1
Danish	80	42	81	269	640	65	149	203	495	628	46
Dutch	1005	597	1002	2346	2401	2444	939	3727	4892	2115	37
Finnish	277	187	258	441	749	558	254	726	1126	872	30
French	300	380	1106	1181	1059	551	480	1431	2255	1371	1
Galician	75	51	248	160	209	77	95	74	641	200	16
German	1698	1815	2416	5313	6289	2725	2082	5346	10298	6694	125
Italian	86	41	59	103	399	77	62	112	432	283	0
Norwegian	952	539	596	2078	3908	1295	613	1496	5030	3455	6
Portuguese	123	144	331	527	688	110	162	334	1111	640	12
Spanish	642	942	1614	3611	3410	549	952	1054	7065	3601	92
Swedish	44	68	129	165	472	165	119	383	419	360	0

Additional data subsetting decisions However, Figure S.5 shows that the longer the sentence, the more quasi-sentences tend to be matched to it. Manual inspection of a sample of sentences that have more than 71 words[10] showed that these long sentences are often enumerations of multiple separate policy statements (hence multiple quasi-sentences). Such long sentences present problems to the sentence embedding models used in my analyses because of their limits to the maximum number of tokens in the input sentence. If not stated otherwise, I have thus discarded sentences with more than 71 words from the sentences matched to 1–6 quasi-sentences with a single unambiguous coding. Applying this cut-off reduces the number of sentences with unambiguous topic coding from 72485 to 71576 and from 72597 to 71661 in sentences with unambiguous position coding.

A final decision was to keep only the 92.13% of sentences that are unambiguously coded on both outcome variable of interest (see Table S.8). This allows using the same cross-validation folds for classifier training across outcomes, which, in turn, allows direct comparison of classifier performances between tasks. Table S.9 reports the distribution of labels across languages in the set of retained sentences.

[10] I have selected this cutoff because it is the 90%-percentile value of numbers of words in sentences that are matched to three quasi-sentences (the 95%-percentile value of numbers of matched quasi-sentences) with an unambiguous position or domain coding.

Table S.10. Number of labeled quasi-sentences and label class proportions by language in Lehmann and Zobel (ibid.) dataset.

		Immigration/Integration issue	
Language	*N*	"Yes"	"No"
Danish	6239	0.09	0.91
Dutch	30931	0.04	0.96
English	30080	0.01	0.99
Finnish	7888	0.03	0.97
German	66135	0.05	0.95
Norwegian	33544	0.04	0.96
Spanish	40074	0.02	0.98
Swedish	7956	0.05	0.95

C.3 Immigration issue codings provided by Lehmann and Zobel (2018)

Lehmann and Zobel (ibid.) have crowd-sourced immigration/integration issue codings for a sample of 223 manifestos from 14 different countries Lehmann and Zobel (ibid.). Their data covers manifestos written in eight different languages and is freely available for download.[11]

To match the immigration/integration codings in their data, I have adapted an `R` script written by the authors.[12] To create the binary outcome indicator, I rely on variables `selection`, `gs_answer_1r`, `topic`, and `gs_answer_2q` in their dataset and have coded all quasi-sentences as immigration/integration instances ("positive" label class) if (i) the gold-standard coding is "immigration" or "integration" , or (ii) if the (majority) crowd coding is "immigration" or "integration." All other quasi sentences have been assigned to the "negative" label class. Table S.10 reports the number of quasi-sentences and label distributions by language.

[11] https://manifesto-project.wzb.eu/down/datasets/pimpo
[12] https://manifesto-project.wzb.eu/down/datasets/pimpo/create_PImPo_with_verbatim.r

Table S.11. Numbers of training samples by language and training split used in Analaysis 1.

		Train–val split				
Language	n_{total}	%	%	%	%	%
Catalan	1765	2.5	2.5	2.5	2.5	2.5
Danish	1372	1.9	1.9	1.9	1.9	1.9
Dutch	10771	15.2	15.2	15.2	15.2	15.2
Finnish	2754	3.9	3.9	3.9	3.9	3.9
French	5058	7.1	7.1	7.1	7.1	7.1
Galician	931	1.3	1.3	1.3	1.3	1.3
German	22463	31.6	31.6	31.6	31.6	31.6
Italian	827	1.2	1.2	1.2	1.2	1.2
Norwegian	9987	14.1	14.1	14.1	14.1	14.1
Portuguese	2097	3.0	3.0	3.0	3.0	3.0
Spanish	11812	16.6	16.6	16.6	16.6	16.6
Swedish	1162	1.6	1.6	1.6	1.6	1.6

D Classifier details

D.1 Training data and cross-validation fold sampling

While the training data used differ across my analyses, in each analysis, I have first sampled n_{train} (quasi-)sentences into the training split. Held-out (quasi-)sentences served as validation samples. Training data sampling strategies vary across analyses, however.

Analyses 1

The first analysis of my study is based on the sentence-level dataset constructed from machine-translated manifesto full texts provided by Düpont and Rachuj (2022). The resulting corpus records 70999 sentences. I have randomly sample these sentences *five times* into 50:50 training and validation data splits. Training datasets thus record 35496‘ sentences.

I have sampled 50% of sentences from each language into a training data split. Table S.11 reports the composition of the training data split.

In these training data splits, I have randomly assigned samples five times into five (80:20%) train–test splits to create cross validation (CV) folds. Lists recording assignments of sentences into data splits respectively CV folds are available on request.

Analysis 2

Like the first analysis, the second analysis of my study is based on the sentence-level dataset constructed from machine-translated manifesto full texts provided by Düpont and Rachuj (ibid.). To estimate classifiers out-of-sample performance as a function of the amount of labeled data available at training time, I have sampled five times 5%, 10%, . . . , 45% of sentences from each language into five different training data splits. Note for

Table S.12. Composition of down-sampled training data split obtained from Lehmann and Zobel (ibid.) data.

	Immigration/Integration issue	
Language	"Yes"	"No"
Danish	352	352
Dutch	829	829
English	258	258
Finnish	162	162
German	1980	1980
Norwegian	797	797
Spanish	579	579
Swedish	240	240

reference that the classifiers in the first analysis have been trained on 50% of the entire corpus. Consequently, the five training datasets representing 5% of labeled sentences in the Düpont and Rachuj (2022) dataset record 3549, those representing 10% 7099, and so on up to 31948 labeled sentences.

To ensure comparability between models trained using different amounts of labeled data, the training data splits have been compiled incrementally by adding labeled sentences in 5 percentage point steps (instead of independently sampling the respective percentage of labeled sentences). As a consequence of this sampling strategy, all samples in the 5% training split are contained in the larger training data splits, etc. I have applied the same logic to construct cross-validation folds.

Analysis 3

The data collected by Lehmann and Zobel (2018) is very imbalanced in the binary immigration/integration issue indicator: 8666 immigration/integration issue instances are outnumbered by more than 214 thousand "negative" instances.

Baseline classifier Hence, when training the baseline classifier reported first in section 4.3 of the main paper, I have down-sampled the training data so that positive and negative labels are balanced. Specifically, I have stratified the data by language and sampled 60% of immigration/integration issue instances and an equal amount of negative instances into the training data split. The exact numbers of training samples by language are reported in Table S.12.

In this training data split, I have randomly assigned samples five times into five (80:20%) train–test splits to create cross-validation (CV) folds. Lists recording assignments of quasi-sentences into data splits respectively CV folds are available on request.

Cross-lingual transfer classifiers In the cross-lingual transfer experiment, in needed to avoid language-imbalance in the training data to ensure that performance differences in classifying held-out sentences can be attributed to the particular combination of languages used for training.

Hence, I have first determined the language with the least number of positive labels: Finnish with 270 positive labels. I have then sampled for each language the same amount of training data: 216 positive and 432 negative samples. (I have doubled the number of negative samples to ensure that there was not too little training data.) Because I have always trained on samples from six different language, the (language-balanced) training datasets thus always comprised 3888 quasi-sentences.

D.2 Classifier training and evaluation

Supervised learning from multilingual sentence embeddings

Supervised learning from MSEs can be implemented as follows: First, texts written in different languages are projected into a joint vector space using a pre-trained MSE model. Second, texts' language-independent representations in this space are used as features to train a supervised machine learning algorithm to obtain a cross-lingual text classifier.

Specifically, given a multilingual corpus $\mathcal{C}$ of $s = 1, \ldots, n$ sentences (or similar sentence-like text sequences), a label $y_s \in \mathcal{Y}$ for each sentence, and a pre-trained MSE model M, *supervised learning from MSEs* is a three-step procedure:

1. *MSE*

   Sentences written in different languages are projected into a joint, language-independent vector space using M. This produces one vector $\mathbf{e}_s \in \mathbb{R}^d$ per sentence ("sentence embedding").

2. *Constructing the feature matrix*

   The sentence embeddings obtained in step (1) are stacked up row-wise in an "embeddings matrix" $\{\mathbf{e}_s^\intercal\}_{s \in \mathcal{C}} = \mathbf{E}_{n \times d}$, with rows recording sentence embeddings and columns embedding dimensions.

3. *Supervised learning*

   The columns of the embedding matrix obtained in step (2) are used as features to train a supervised classifier using labels $(y_s)_{s=1,\ldots,n}$ as outputs.

Embedding To embed the texts of (quasi-)sentences in training and validation data splits, I have used the LASER encoder, the knowledge-distilled mUSE model and the knowledge-distilled XLM-R model as described in listings B.4, B.2, respectivelyB.2 (see section B). Since all three models return one vector per input text, this resulted in three matrices with dimensions $n_{\text{train}} \times l$ and $n_{\text{val}} \times l$, where $l = 1024$ for the LASER, $l = 512$ for the knowledge-distilled mUSE model, and $l = 768$ for the knowledge-distilled XLM-R model. (Quasi-)Sentences' representations in the embedding space were then used as features to train supervised learning algorithms. Note that for topic or position classification, I have always included the 'uncoded' category as an additional outcome label.

Classifier training I have trained L2-regularized generalized linear models (GLM) to the training data splits in all analyses. My algorithm choice is motivated by both practical and methodological considerations: A linear classifier is conceptually simple, most readers in the political and communication sciences should be familiar with GLMs, and there exists a fast R implementation of the GLM (`glmnet`) that has been developed by professional statisticians. From a methodological point of view, L1-regularization was unlikely to yield good results as embeddings are distributed representations since it allows regression parameters to be zero.

The strength of L2-regularization of the GLM is governed by the hyper-parameter λ, with higher values indicating more regularization. I have identified optimal λ values using 5-times repeated 5-fold cross-validation (5x5 CV) in the training data (hyper-parameter tuning). As candidate values of λ I have examined a grid of 8 values including 0 (i.e., no regularization) and 7 values of $1/10^x$ with $x \in \{1, 1.33, 1.66, \dots, 3\}$.

The "best" model among classifiers evaluated during CV was determined based on the cross-CV average of the cross-class mean F1 score, and the hyperparameter values of this model were used to train the algorithm to all (quasi-)sentences in the training data split. The resulting classifier was then evaluated on held-out validation samples to compute estimates of out-of-sample predictive performance.

Supervised learning from bag-of-words representations of machine-translated texts

In the first two analyses, I have used the sentence-level dataset of machine-translated manifestos described in section C.2 to compare supervised learning from MSEs to the current state-of-the-art in political text classification: supervised learning from bag-of-words representations of machine-translated full texts. To implement supervised learning from translated texts' bag-of-words (BoW) representations, I have relied on sentences' machine-translated English versions. Specifically, I have relied on the original sentence-level translations Düpont and Rachuj have obtained using *Google Tranlsate*. In addition, I have translated the same sentences into English using the open-source M2M model (Fan et al., 2021).

In the third analysis, I have then relied only on M2M for translation because the

Table S.13. Default text pre-processing steps applied to manifesto sentences' English translations.

Step	Description
tokenizing	texts are tokenized using ICU-based word boundary splitting (see http://userguide.icu-project.org/boundaryanalysis)
punctuation removed	punctation is removed using the unicode `P` class
numbers removed	standalone numbers are removed
symbols removed	punctation is removed using the unicode `S` class
separators removed	separators are removed using the unicode `Z` and `C` classes
hyphens splited	hypens are splitted

Table S.14. Additional text pre-processing steps applied to manifesto sentences' English translations as part of cross-validation.

Variant	Description
1	stemming, stop word removal, unigrams only
2	stemming, unigrams only, (.01, .9) document frequency based-trimming
3	stemming, uni-, bi- and trigrams, (.01, .9) document frequency based-trimming
4	stemming, stop word removal, uni-, bi- and trigrams, (.01, .9) document frequency based-trimming

dataset compiled by Lehmann and Zobel (2018) is based on the original CMP quasi-sentence level data and thus records no machine translations.

Text preprocessing To obtain BoW representations from machine-translated texts, I have first *always* applied the tokenization and pre-processing steps listed in Table S.13 to texts in the training data split. Because BoW classifiers' performance can depend on how texts are pre-processed, I have also varied the pre-processing steps ("variants") listed in Table S.14 as part of the cross-validation procedure. That is, the variants listed in Table S.14 have been treated as hyper-parameter choices while "default" pre-processing steps have been applied invariantly. In addition, I have used the resulting BoW representations always once as-is (i.e., count-vectorized) and once after transforming them by applying `tf-idf` reweighting. For each task (topic and position classification), this resulted in a total of eight feature representations.

Classifier training

Using these data, For each task, I have then selected the "best" performing combination of learning algorithm and BoW feature representation based on classifiers' average cross-class F1 scores in 5x5 CV. Note that I have used the same CV folds as when training the embedding-based classifiers.

D.3 Setup of the cross-lingual classification experiment

As described above, for each of the eight languages in the Lehmann and Zobel (2018) dataset, I have first created five 80:20 training–validation splits with 648 quasi-sentences in the training data split of each language. Combining six languages' training data splits, I have then trained one MSE-based classifiers (relying on the pre-trained XLM-R model) and one MT-based classifier (using M2M for translation). Because I have sample five different training data splits for each language, training two classifiers per training data set results in a total of 10 classifiers per source language set.

Next, I have evaluated these classifiers on *all* languages' corresponding validation datasets, bootstrapping 50 F1 scores per language. For each evaluation, these F1 score estimates stem from out-of-language classification for two languages and from within-language classification for the other six languages.

I repeat this procedure for all 28 possible combinations of source languages (i.e., source language sets) and the five training–validation splits I obtaind by combining languages respective training and validation data splits. This results in the 56000 bootstrapped F1 estimates (1120 evaluations) per task I summarize in Figure 4 in the main paper.

Moreover, I have use these data to quantifying the "reliability cost" of cross-lingual transfer for each approach and target language by subtracting the mean F1 score achieved in within-language evaluation from that achieved in out-of-language evaluation. Negative signs on these estimates indicate that cross-lingual transfer results, on average, in less reliable classification compared to the within-language classification benchmark and their magnitude quantifies by how much.

Table S.15. Average class-wise F1 of classifiers presented in Figure 2 in the main paper.

class	commercial (Google)	free (M2M)	LASER	mUSE	XLM-R
Topic classification					
econ	0.62	0.61	0.61	0.62	0.64
extrel	0.60	0.59	0.62	0.61	0.65
fabsoc	0.53	0.52	0.50	0.53	0.55
freedem	0.49	0.48	0.48	0.49	0.52
polsys	0.46	0.46	0.45	0.46	0.48
socgrp	0.41	0.40	0.37	0.40	0.42
welqual	0.62	0.61	0.62	0.63	0.65
Position classification					
left	0.52	0.51	0.47	0.48	0.49
neutral	0.67	0.67	0.70	0.71	0.72
right	0.59	0.59	0.54	0.57	0.59

E Additional results

E.1 Analysis 1: Comparative reliability

Figure S.6 reports the cross-class mean F1 scores of classifiers presented in section 4.1 of the main paper. Table S.15 reports their class-wise F1 scores. Figure S.7 reports their language-specific (cross-class mean) F1 scores. Figure S.8 reports the distribution of differences in the XLM-R and Google MT-based classifiers' F1 scores by language.

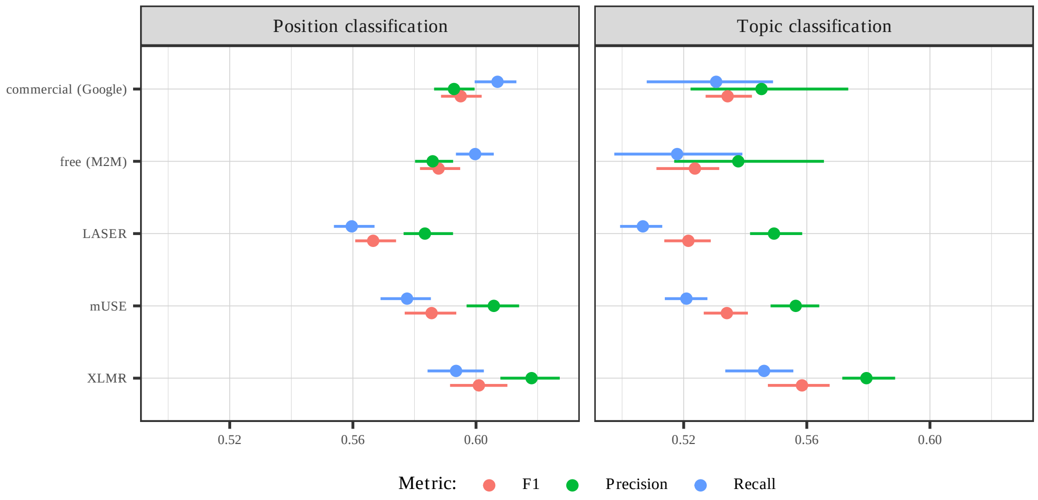

Figure S.6. Cross-class mean F1, precision, and recall of classifiers presented in Figure 2 in the main paper. Data plotted summarizes 50 bootstrapped cross-class mean estimates (excluding the `uncoded` category) for five classifiers per task and approach. Points are averages of bootstrapped estimates and vertical lines span the 95% most frequent values.

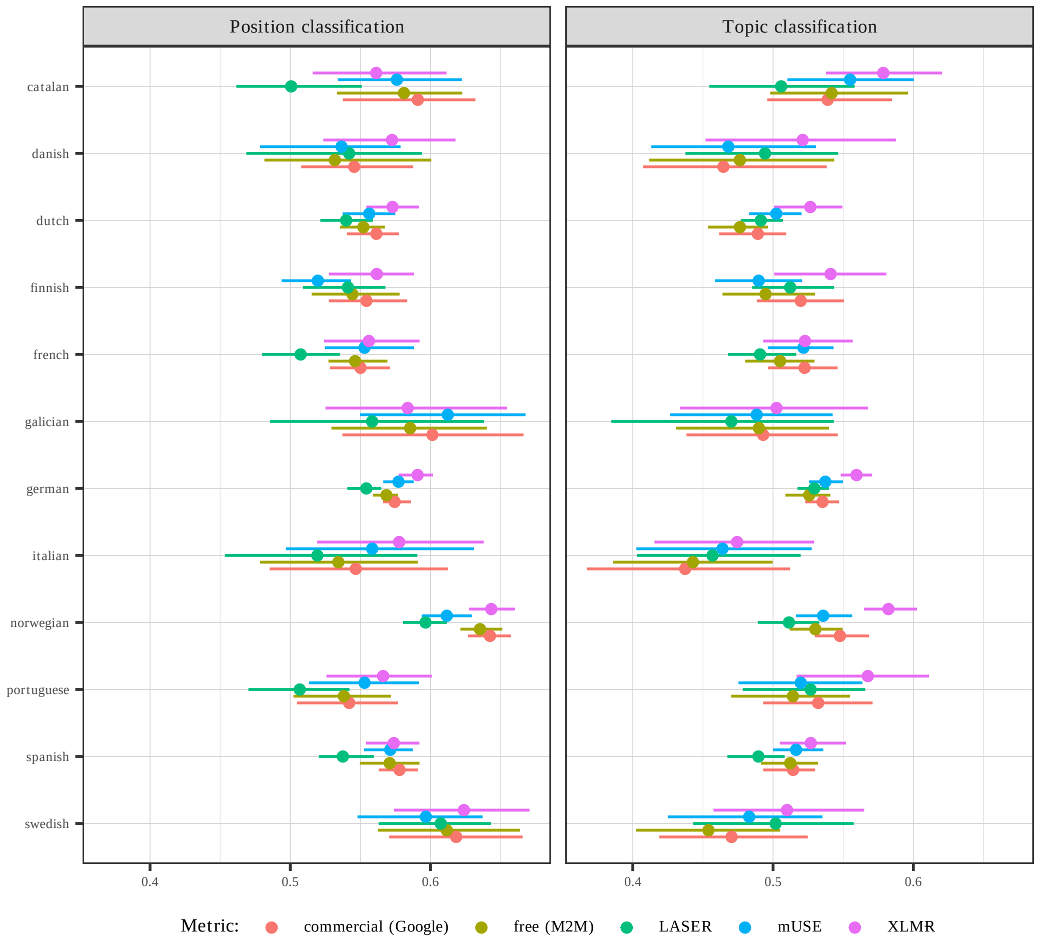

Figure S.7. Cross-class mean F1 by language of classifiers presented in Figure 2 in the main paper. Data plotted summarizes 50 bootstrapped cross-class mean F1 scores (excluding the `uncoded` category) for five classifiers per task, approach, and language. Points are averages of bootstrapped estimates and vertical lines span the 95% most frequent values.

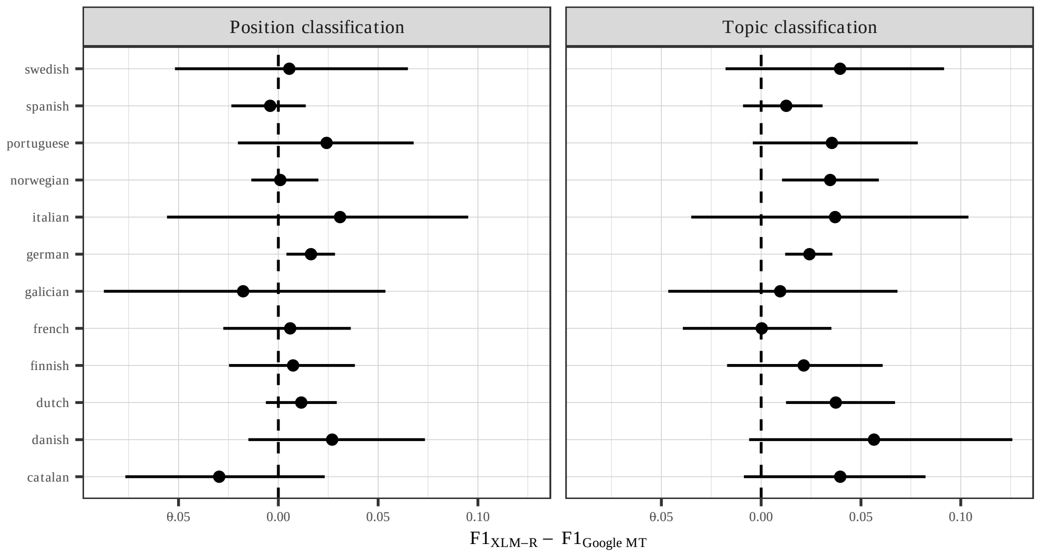

Figure S.8. Differences in language-specific cross-class mean F1 scores of XLM-R and Google MT-based classifiers presented in Figure 2 in the main paper. Data plotted summarizes differences in 50 bootstrapped cross-class mean F1 scores (excluding the `uncoded` category) for five classifiers per task, approach, and language. Points are average averages of differences in bootstrapped estimates and vertical lines span the 95% most frequent values.

F References

Artetxe, M. and H. Schwenk (2019). “Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond”. In: *Transactions of the Association for Computational Linguistics* 7, pp. 597–610.

Cer, D., Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil (2018). “Universal Sentence Encoder for English”. In: *2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.* Association for Computational Linguistics, pp. 169–174.

Chidambaram, M., Y. Yang, D. Cer, S. Yuan, Y. Sung, B. Strope, and R. Kurzweil (2019). “Learning Cross-Lingual Sentence Representations via a Multi-Task Dual-Encoder Model”. In: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019).* Association for Computational Linguistics, pp. 250–259.

Conneau, A., K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán,. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov (2020). “Unsupervised Cross-lingual Representation Learning at Scale”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451.

Düpont, N. and M. Rachuj (2022). “The Ties That Bind: Text Similarities and Conditional Diffusion among Parties”. In: *British Journal of Political Science* 52.2, pp. 613–630.

Fan, A., S. Bhosale, H. Schwenk, Z. Ma, A. El-Kishky, S. Goyal, M. Baines, O. Celebi, G. Wenzek, and V. Chaudhary (2021). “Beyond English-Centric Multilingual Machine Translation”. In: *Journal of Machine Learning Research* 22.107, pp. 1–48.

Lample, G. and A. Conneau (2019). “Cross-Lingual Language Model Pretraining”.

Lehmann, P. and M. Zobel (2018). “Positions and Saliency of Immigration in Party Manifestos: A Novel Dataset Using Crowd Coding”. In: *European Journal of Political Research* 57.4, pp. 1056–1083.

Lewandowski, J., N. Merz, S. Regel, P. Lehmann, and P. Muscat (2020). *manifestoR: Access and Process Data and Documents of the Manifesto Project.*

Reimers, N. (2021). *Sentence-Transformers (Version 0.4.1).*

Reimers, N. and I. Gurevych (2019). “Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992.

— (2020). “Making Monolingual Sentence Embeddings Multilingual Using Knowledge Distillation”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing.* Empirical Methods in Natural Language Processing (EMNLP), pp. 4512–4525.

Schwenk, H. and M. Douze (2017). “Learning Joint Multilingual Sentence Representations with Neural Machine Translation”. In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pp. 157–167.

Vaginay, Y. (2020). *Laserembeddings (Version 1.1.0).*

Volkens, A., T. Burst, W. Krause, P. Lehmann, T. MatthieSS, N. Merz, S. Regel, B. WeSSels, L. Zehnter, and W. B.F. S. (WZB) (2020). *Manifesto Project Dataset.* Version 2020a.

Werner, A., O. Lacewell, and A. Volkens (2015). *Manifesto Coding Instructions: 5th Fully Revised Edition.*

Yang, Y., D. Cer, A. Ahmad, M. Guo, J. Law, N. Constant, G. H. Abrego, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil (2020). “Multilingual Universal Sentence Encoder for Semantic Retrieval”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 87–94.