Heterogeneous Document Embeddings for Cross-Lingual Text Classification

Alejandro Moreo
alejandro.moreo@isti.cnr.it
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy

Andrea Pedrotti
andrea.pedrotti@phd.unipi.it
Dipartimento di Informatica
Università di Pisa
56127 Pisa, Italy

Fabrizio Sebastiani
fabrizio.sebastiani@isti.cnr.it
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy

ABSTRACT

Funnelling (Fun) is a method for *cross-lingual text classification* (CLC) based on a two-tier ensemble for heterogeneous transfer learning. In Fun, 1st-tier classifiers, each working on a different, language-dependent feature space, return a vector of calibrated posterior probabilities (with one dimension for each class) for each document, and the final classification decision is taken by a metaclassifier that uses this vector as its input. The metaclassifier can thus exploit class-class correlations, and this (among other things) gives Fun an edge over CLC systems where these correlations cannot be leveraged.

We here describe *Generalized Funnelling* (gFun), a learning ensemble where the metaclassifier receives as input the above vector of calibrated posterior probabilities, concatenated with document embeddings (aligned across languages) that embody other types of correlations, such as word-class correlations (as encoded by *Word-Class Embeddings*) and word-word correlations (as encoded by *Multilingual Unsupervised or Supervised Embeddings*). We show that gFun improves on Fun by describing experiments on two large, standard multilingual datasets for multi-label text classification.

KEYWORDS

Heterogeneous Transfer Learning, Transfer Learning, Text Classification, Ensemble Learning, Word Embeddings

ACM Reference Format:
Alejandro Moreo, Andrea Pedrotti, and Fabrizio Sebastiani. 2021. Heterogeneous Document Embeddings for Cross-Lingual Text Classification. In *SAC 2021: 36th ACM/SIGAPP Symposium On Applied Computing, March 22–26, 2021, Gwangju, South Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/XX.XXXX/XXXXX.XXXX

1 INTRODUCTION

Transfer Learning (TL) is a class of machine learning tasks in which, given a training set $\mathrm{Tr}^L_{\mathcal{S}}$ of labelled data items from a "source" domain $\mathcal{S}$, we must issue predictions for unlabelled data items from a "target" domain $\mathcal{T}$, related to $\mathcal{S}$ but different from it. *Heterogeneous* TL (HTL) denotes the set of TL tasks in which the feature spaces $F_{\mathcal{S}}$ and $F_{\mathcal{T}}$ of the two domains are different (and, in general, non-overlapping). An example HTL task is *cross-lingual text classification* (CLC), the task of classifying documents, each written in one of a finite set $\mathcal{L} = \{\lambda_1, ..., \lambda_{|\mathcal{L}|}\}$ of languages, according to a common "codeframe" (or: classification scheme) $\mathcal{Y} = \{y_1, ..., y_{|\mathcal{Y}|}\}$. CLC can be tackled as a TL task, with the goal of improving on the naïve "monolingual baseline" (consisting of $|\mathcal{L}|$ independently generated monolingual classifiers) by exploiting synergies among training sets from different languages. Here, each language-specific domain of documents is at the same time a source domain and a target domain, according to an "all languages help each other" metaphor.

Esuli et al. [3] proposed *Funnelling* (Fun), a two-tier ensemble method for HTL, and tested it on a CLC setting. In Fun, a set of $|\mathcal{L}|$ 1st-tier, language-specific classifiers return, for each unlabelled document d, a vector of $|\mathcal{Y}|$ calibrated posterior probabilities; each such vector is fed to a 2nd-tier "metaclassifier" which returns the final classification decisions. Vectors of $|\mathcal{Y}|$ calibrated posterior probabilities thus form an "interlingua" among the $|\mathcal{L}|$ languages, since all such vectors are in the same vector space, irrespectively of the language of the documents they correspond to.

One of the reasons Fun outperforms the naïve monolingual baseline is that the metaclassifier leverages *class-class correlations*, i.e., stochastic dependencies among the different classes in $\mathcal{Y}$. In this paper we propose *Generalized Funnelling* (gFun), an extension of Fun capable of leveraging additional types of correlations (e.g., word-class correlations, word-word correlations). This is obtained by aggregating the vector of calibrated posterior probabilities and document embeddings that encode these additional types of correlations. We here present CLC experiments in which we extend the vectors of calibrated posterior probabilities by using *Word-Class Embeddings* (WCEs) [6] and *Multilingual Unsupervised or Supervised Embeddings* (MUSEs) [2], which encode word-class correlations and word-word correlations for multiple languages, respectively.

We describe gFun in §2. In §3 we report on experiments we have performed on two large multilingual datasets for multi-label text classification. In §4 we conclude by sketching avenues for further research. Our code that implements gFun is publicly available.[1]

2 GENERALIZED FUNNELLING

2.1 A Brief Introduction to Funnelling

Funnelling, as described in [3], comes in two variants, called Fun(kfcv) and Fun(tat); we here disregard Fun(kfcv) and only use Fun(tat),

[1] https://github.com/andreapdr/gFun

since in all the experiments reported in [3], Fun(tat) clearly outperformed Fun(kfcv). Both Fun and gFun can tackle single-label and multi-label text classification alike; for reasons of space, we here deal only with the latter.

In Fun(tat), in order to train a classifier ensemble, we first train language-specific, 1st-tier classifiers $h^1_1, ..., h^1_{|\mathcal{L}|}$ (with superscript s indicating the s-th tier) from the language-specific training sets $\mathrm{Tr}_1, ..., \mathrm{Tr}_{|\mathcal{L}|}$. Training documents $d \in \mathrm{Tr}_i$ may be represented by means of any desired vectorial representation $\phi^1(d)$, such as, e.g., *tfidf*-weighted bag-of-words, and classifiers may be trained by any learner, provided the resulting classifier returns, for each document d to classify and for each class y_j, a confidence score $h^1_i(d, y_j) \in \mathbb{R}$, where λ_i is the language document d is written in.

We then apply each 1st-tier classifier h^1_i to all training documents $d \in \mathrm{Tr}_i$, thus obtaining a vector

$$S(d) = (h^1_i(d, y_1), ..., h^1_i(d, y_{|\mathcal{Y}|})) \tag{1}$$

of confidence scores for each $d \in \mathrm{Tr}_i$.

The next step consists of computing (via a chosen probability calibration method) language- and class-specific calibration functions f_{ij} that map confidence scores $h^1_i(d, y_j)$ into calibrated posterior probabilities. We can then apply f_{ij} to each document $d \in \mathrm{Tr}_i$ and obtain a vector of calibrated posterior probabilities

$$\begin{aligned}\phi^2(d) &= (f_{i1}(h^1_i(d, y_1)), ..., f_{i|\mathcal{Y}|}(h^1_i(d, y_{|\mathcal{Y}|}))) \\ &= (\Pr(y_1|d), ..., \Pr(y_{|\mathcal{Y}|}|d))\end{aligned} \tag{2}$$

At this point, we train a language-independent, 2nd-tier "meta"-classifier h^2 from all training documents $d \in \bigcup_{i=1}^{|\mathcal{L}|} \mathrm{Tr}_i$, where document d is represented by its $\phi^2(d)$ vector. This concludes the training phase.

In order to apply the trained ensemble to a test document $d \in \mathrm{Te}_i$ we apply classifier h^1_i to d and convert the resulting vector $S(d)$ of confidence scores into a vector $\phi^2(d)$ of calibrated posterior probabilities. We then feed this latter into the metaclassifier h^2, which returns a vector of confidence scores $(h^2(d, y_1), ..., h^2(d, y_{|\mathcal{Y}|}))$ from which the final decisions are obtained in the usual way.

2.2 Introducing Heterogeneous Correlations through Generalized Funnelling

As explained in [3], the reasons of the good performance of Fun are essentially two. The first is that Fun learns from heterogeneous data; i.e., while in the naïve monolingual baseline each classifier is trained on just $|\mathrm{Tr}_i|$ labelled examples, in Fun we have a metaclassifier trained on $\bigcup_{i=1}^{|\mathcal{L}|} |\mathrm{Tr}_i|$ labelled examples, which means that all training examples contribute to classifying all unlabelled examples, irrespectively of the languages of the former and of the latter. The second is that the metaclassifier leverages *class-class correlations*, i.e., it learns to exploit the stochastic dependencies between classes typical of multilabel settings.

The goal of gFun is that of allowing additional types of stochastic dependencies (e.g., word-class correlations, word-word correlations) to contribute to the classification process.

The key step in allowing Fun's metaclassifier to leverage the different language-specific training sets consists of representing their documents in a space that is common to all languages. In Fun, this is made possible by the fact that the 1st-tier classifiers all return vectors of calibrated posterior probabilities. In gFun (Algorithm 1) this process is generalized by introducing a set Ψ of *view generators*, i.e., language-dependent functions mapping documents into language-independent vectorial representations *aligned across languages*, i.e., such that both the dimensionality of the vectors and the meaning of each vector dimension are the same for all languages.

```
Input  : • Sets {$\mathrm{Tr}_1, ..., \mathrm{Tr}_{|\mathcal{L}|}$} of training documents written in languages
           $\mathcal{L} = \{\lambda_1, ..., \lambda_{|\mathcal{L}|}\}$, all labelled according to $\mathcal{Y} = \{y_1, ..., y_{|\mathcal{Y}|}\}$;
         • Sets {$\mathrm{Te}_1, ..., \mathrm{Te}_{|\mathcal{L}|}$} of unlabelled documents written in languages
           $\mathcal{L} = \{\lambda_1, ..., \lambda_{|\mathcal{L}|}\}$, all to be labelled according to
           $\mathcal{Y} = \{y_1, ..., y_{|\mathcal{Y}|}\}$;
         • Set $\Psi = \{\psi_1, ..., \psi_{|\Psi|}\}$ of view generators;
Output : • Trained gFun architecture ;
         • Labels for all documents in {$\mathrm{Te}_1, ..., \mathrm{Te}_{|\mathcal{L}|}$} ;
1  /* Training phase */
2  for $\lambda_i \in \mathcal{L}$ do
3      /* Learn the parameters of each view generator */
4      for $\psi_k \in \Psi$ do
5          $\theta_{ik} \leftarrow fit(\psi_k, \mathrm{Tr}_i)$ ;
6      end
7      /* Generate 1st-tier language-independent views */
8      $\mathrm{Tr}'_i \leftarrow (\psi_1(\mathrm{Tr}_i, \theta_{i1}), \ldots, \psi_{|\Psi|}(\mathrm{Tr}_i, \theta_{i|\Psi|}))$ ;
9      /* Aggregate the language-independent views */
10     $\mathrm{Tr}''_i \leftarrow aggfunc(\mathrm{Tr}'_i)$ ;
11 end
12 /* Combine all training sets */
13 $\mathrm{Tr} \leftarrow \bigcup_{i=1}^{|\mathcal{L}|} \mathrm{Tr}''_i$ ;
14 Train classifier $h^2$ from all vectors in Tr ;
15 /* Classification phase */
16 for $\lambda_i \in \mathcal{L}$ do
17     /* Generate 1st-tier language-independent views */
18     $\mathrm{Te}'_i \leftarrow (\psi_1(\mathrm{Te}_i, \theta_{i1}), \ldots, \psi_{|\Psi|}(\mathrm{Te}_i, \theta_{i|\Psi|}))$ ;
19     /* Aggregate the language-independent views */
20     $\mathrm{Te}''_i \leftarrow aggfunc(\mathrm{Te}'_i)$ ;
21     /* Invoke the meta-classifier */
22     Apply $h^2$ to all vectors in $\mathrm{Te}''_i$ ;
23 end
```

Algorithm 1: Generalized Funnelling for CLC.

The view generators $\psi_k \in \Psi$ might require parameter optimization during the training phase; this is undertaken in Line 5, independently for each language and view generator. gFun also implements an aggregation function (*aggfunc* – Line 10) that brings together the different representations produced by the view generators, and shapes the document representation for use in the metaclassifier. In this case, as *aggfunc* we simply adopt concatenation.

Note that the original formulation of Fun (Section 2.1) thus reduces to an instantiation of gFun in which there is a single view generator (a calibrated classifier) and the aggregation function is the identity function. In this case, fitting this single view generator comes down to training the 1st-tier classifier h^1_i and choosing the calibration functions f_{ik}. During the test phase, invoking the view generator (Line 18) amounts to computing the $\phi^2(d)$ representations (Equation 2) of the test documents.

The Fun metaclassifier has access to vectors of $|\mathcal{Y}|$ posterior probabilities, and can thus leverage class-class correlations. In what follows we instead describe new view generators that we have investigated in order to introduce additional information into gFun. In particular, we describe view generators that mine word-class correlations (Section 2.2.1) and word-word correlations (Section 2.2.2).

We also discuss a few additional modifications concerning data normalization (Section 2.2.3) that we have introduced into gFun and that, although subtle, bring about a substantial improvement in the effectiveness of the method.

2.2.1 Word-Class Correlations. For encoding word-class correlations we derive document embeddings from *Word-Class Embeddings* (WCEs) [6]. WCEs are supervised embeddings meant to extend (e.g., by concatenation) other unsupervised pre-trained word embeddings (e.g., those produced by `word2vec` or `GloVe`) in order to inject task-specific word meaning in multiclass text classification. The WCE for word w is defined as

$$E(w) = \varphi(\eta(w, y_1), ..., \eta(w, y_{|\mathcal{Y}|})) \tag{3}$$

where η is a real-valued function that quantifies the correlation between word w and class y_j as observed in the training set, and where φ is any dimensionality reduction function. Here, as the η function we adopt the normalized dot product, as proposed in [6], whose computation is very efficient; as φ we use the identity function, and our WCEs are thus $|\mathcal{Y}|$-dimensional vectors.

So far, WCEs have been tested exclusively in monolingual settings. However, WCEs are *naturally aligned across languages*, since WCEs have one dimension for each $y \in \mathcal{Y}$, which is the same for all $\lambda_i \in \mathcal{L}$. Document embeddings relying on WCEs thus display similar characteristics irrespective of the language in which the document is written in. This is, to the best of our knowledge, the first application of WCEs to a multilingual setting.

The view generator for WCEs consists of first computing, for each language $\lambda_i \in \mathcal{L}$, the language-specific WCE matrix $\mathbf{W}_i$ (Line 5), and then projecting the *tfidf* matrix $\mathbf{X}_i$ of Tr_i (during training – Line 8) or Te_i (during test – Line 18) as $\mathbf{X}_i \cdot \mathbf{W}_i$.

2.2.2 Word-Word Correlations. For encoding word-word correlations we derive document embeddings from *Multilingual Unsupervised or Supervised Embeddings* (MUSEs) [2]. MUSEs are generated via a method for aligning in a common vector space unsupervised (monolingual) word embeddings. The alignment is obtained via a linear mapping (i.e., a rotation matrix) W learned by an adversarial training process in which a *generator* (in charge of mapping the source embeddings onto the target space) is trained to fool a *discriminator* from distinguishing the language of provenance of the embeddings, that is, from discerning if the embeddings it receives as input originate from the target language or are instead the product of a transformation of embeddings originated from the source language. Mapping W is then further refined using Procrustes alignment. The name "Unsupervised or Supervised" refers to the fact that the method can operate with or without a dictionary of parallel seed words.

We used the MUSEs that the authors of [2] make publicly available[2], and that consist of 300-dimensional multilingual word embeddings trained on Wikipedia using fastText. The embeddings have been aligned for 30 languages with the aid of a bilingual dictionary.

The view generator for MUSEs is similar to that for WCEs, with the sole exception that fitting the generator (Line 5) comes down to just allocating in memory the pre-trained MUSE matrices $\mathbf{M}_i$ for each language λ_i involved; the projection of training and test documents is as before, and is computed as $\mathbf{X}_i \cdot \mathbf{M}_i$.

[2]https://github.com/facebookresearch/MUSE

Table 1: Characteristics of the datasets used in [3] and in this paper; $|\mathcal{L}|$, $|\mathcal{Y}|$, $|\mathrm{Tr}|$, and $|\mathrm{Te}|$ indicate the number of languages, classes, training documents, and test document used in each of the 10 runs; CDOC indicates the average number of classes per document, which is the same for all 10 runs. Sets $\mathcal{L}$ and $\mathcal{Y}$ are the same for all runs, while sets Tr and Te are different for each of the 10 runs.

	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{Y}\rvert$	$\lvert\mathrm{Tr}\rvert$	$\lvert\mathrm{Te}\rvert$	CDOC
RCV1/RCV2	9	73	9,000	8,794	3.21
JRC-Acquis	11	300	12,687	46,662	3.31

2.2.3 Normalization. We have found that applying some routine normalization techniques consistently increases the performance of gFun. This normalization consists of imposing unit L2-norm to the vectors computed by the view generators, removing (following [1]) the first principal component of the document embeddings obtained via WCEs or MUSEs, and standardizing the columns of the shared space before passing the vectors to the metaclassifier.[3]

The intuition behind normalization, when dealing with heterogeneous representations, is straightforward, and is that of allowing all sources of information to equally contribute to the classification process. What instead might come as a surprise is the fact that normalization helps improve gFun even when relying exclusively on the class-class correlations (i.e., as Fun does [3]), and that this improvement is statistically significant. We quantify this variation in performance in the experiments of Section 3.

3 EXPERIMENTS

In order to maximize comparability with previous results we adopt an experimental setup identical to the one used in [3], which we briefly sketch here.

Datasets. The first of our two datasets is RCV1/RCV2, a comparable corpus of news stories published by Reuters, while the second is JRC-Acquis, a parallel corpus of legislative texts published by the European Union. We only give some summary characteristics of these datasets in Table 1, and refer the reader to [3] for more details. For both datasets, both the training set and the test set that [3] uses are samples from the original RCV1/RCV2 and JRC-Acquis datasets, and results reported in [3] are averages across 10 runs in which the original datasets are resampled each time;[4] we do the same here.

Evaluation measures. We evaluate classification accuracy via microaveraged F_1 and macroaveraged F_1 (indicated as F_1^{μ} and F_1^{M}, resp.). We also test the statistical significance of differences in performance via paired sample, two-tailed t-tests at the $\alpha = 0.05$ and $\alpha = 0.001$ confidence levels.

Learners. We use the same learner as in [3], i.e., Support Vector Machines (SVMs), as implemented in the `scikit-learn` package.[5]

[3]Standardizing (a.k.a. "z-scoring", or "z-transforming") consists of having a random variable x, with mean μ and standard deviation σ, translated and scaled as $z = \frac{x-\mu}{\sigma}$, so that the new random variable z has zero mean and unit variance. The statistics μ and σ are unknown, and are thus estimated on the training set.

[4]The 10 samples used for each dataset have been made available by the authors of [3] at http://hlt.isti.cnr.it/funnelling/

[5]https://scikit-learn.org/stable/index.html

Table 2: CLC results; each cell indicates the mean value and the standard deviation across the 10 runs. Boldface indicates the best method. Superscripts † and †† denote the method (if any) whose score is not statistically significantly different from the best one; symbol † indicates $0.001 < p$-value< 0.05 while symbol †† indicates $0.05 \leq p$-value.

		Naïve	LRI [5]	CLESA [7]	KCCA [8]	DCI [4]	Fun [3]	gFun(X)	gFun(XM)	gFun(XW)	gFun(XMW)
F_1^μ	RCV1/RCV2	.776 ± .052	.771 ± .050	.714 ± .061	.616 ± .065	.770 ± .052	.802 ± .041	.798 ± .041	**.810** ± .039	.798 ± .042	.809 ± .039††
	JRC-Acquis	.559 ± .012	.594 ± .016	.557 ± .024	.357 ± .023	.510 ± .014	.587 ± .009	.587 ± .010	.601 ± .010	.601 ± .009	**.605** ± .009
F_1^M	RCV1/RCV2	.467 ± .083	.490 ± .077	.471 ± .074	.385 ± .079	.485 ± .070	.534 ± .066	.547 ± .065	.552 ± .063††	.545 ± .064††	**.554** ± .064
	JRC-Acquis	.340 ± .017	.411 ± .027	.379 ± .034	.206 ± .018	.317 ± .012	.399 ± .013	.432 ± .015	.431 ± .016	**.438** ± .012††	**.438** ± .013††

For the 2nd-tier classifier of gFun, and for all the baseline methods, we optimize the C parameter, that trades off between training error and margin, testing all values of $C = 10^i$ for $i \in \{-1, ..., 4\}$ via k-fold cross-validation. We use Platt calibration in order to calibrate the 1st-tier classifiers. We employ the linear kernel for the 1st-tier classifiers and the RBF kernel for the 2nd-tier classifier.

Baselines. As the baselines against which to compare gFun we use the naïve monolingual baseline (hereafter indicated as Naïve), Funnelling (Fun), plus the four best baselines of [3], namely, *Lightweight Random Indexing* (LRI) [5], *Cross-Lingual Explicit Semantic Analysis* (CLESA) [7], *Kernel Canonical Correlation Analysis* (KCCA) [8], and *Distributional Correspondence Indexing* (DCI) [4]. For all systems but gFun, the results we report are excerpted from [3], so we refer to that paper for the detailed setups of these baselines.

Concerning all of the above settings, we stress that they are the settings used in [3], and we adopt them for reasons of comparability.

3.1 Results

Table 2 lists the results obtained via the different methods. For different variants of gFun we indicate in parentheses the document representations that the variant uses, with the vectors of calibrated posterior probabilities denoted by X, and with document embeddings obtained via MUSEs and WCEs denoted by M and W, resp.

gFun(X), the variant that uses the same document representation as Fun, outperforms Fun, which indicates that the normalization steps of Section 2.2.3 are beneficial. The results of gFun(XM) and gFun(XW) show that the M and W representations contribute differently, depending on the nature of the dataset: on RCV1/RCV2, adding M delivers better results than adding W, while W is more useful than M on JRC-Acquis. This can be ascribed to the higher number of classes that JRC-Acquis (300) has with respect to RCV1/RCV2 (73): the 300 classes of JRC-Acquis enable WCEs (that encode word-class correlations) to bring in a higher amount of information, thus making WCEs more discriminative for JRC-Acquis than for RCV1/RCV2. Using all three representations, as in gFun(XMW), yields the best result in 3 out of 4 (measure, dataset) combinations, and in the 4th combination yields a result not different, in a statistically significant sense, from the best one; this confirms the value of the intuitions that underlie gFun.

4 CONCLUSIONS

In this paper we propose an enhancement of the Funnelling ensemble learning method (Fun) [3], called Generalized Funnelling (gFun). We do this by extending Fun's original architecture in order to allow additional sources of information to be brought to bear on the classification process; while Fun leveraged class-class correlations only, we have shown how gFun can also exploit word-word correlations (for which we use MUSE embeddings [2]) and word-class correlations (for which we use WCE embeddings [6]), feeding all these heterogeneous representations to a metaclassifier that, by virtue of their being aligned across languages, can exploit them all. The extensive empirical evaluation that we have carried out confirms that gFun achieves results superior to those obtained by Fun and by a number of well-known baselines. Our current efforts are directed towards testing methods more sophisticated than simple concatenation for combining the three representations, and towards recasting the framework as a deep learning architecture.

ACKNOWLEDGMENTS

The present work has been supported by the ARIADNEplus project, funded by the European Commission (Grant 823914) under the H2020 Programme INFRAIA-2018-1, by the AI4Media project, funded by the European Commission (Grant 951911) under the H2020 Programme ICT-48-2020, and by the SoBigData++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1. The authors' opinions do not necessarily reflect those of the European Commission.

REFERENCES

[1] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*. Toulon, FR.
[2] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*. Vancouver, CA.
[3] Andrea Esuli, Alejandro Moreo, and Fabrizio Sebastiani. 2019. Funnelling: A new ensemble method for heterogeneous transfer learning and its application to cross-lingual text classification. *ACM Transactions on Information Systems* 37, 3 (2019), Article 37. https://doi.org/10.1145/3326065
[4] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. 2016. Distributional correspondence indexing for cross-lingual and cross-domain sentiment classification. *Journal of Artificial Intelligence Research* 55 (2016), 131–163. https://doi.org/10.1613/jair.4762
[5] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. 2016. Lightweight random indexing for polylingual text classification. *Journal of Artificial Intelligence Research* 57 (2016), 151–185. https://doi.org/10.1613/jair.5194
[6] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. 2019. Word-class embeddings for multiclass text classification. (2019). arXiv preprint arXiv:1911.11506.
[7] Philipp Sorg and Philipp Cimiano. 2012. Exploiting Wikipedia for cross-lingual and multilingual information retrieval. *Data and Knowledge Engineering* 74 (2012), 26–45. https://doi.org/10.1016/j.datak.2012.02.003
[8] Alexei Vinokourov, John Shawe-Taylor, and Nello Cristianini. 2002. Inferring a semantic representation of text via cross-language correlation analysis. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems (NIPS 2002)*. Vancouver, CA, 1473–1480.