Generalized Funnelling: Ensemble Learning and Heterogeneous Document Embeddings for Cross-Lingual Text Classification

ALEJANDRO MOREO*, ANDREA PEDROTTI*, and FABRIZIO SEBASTIANI*, Consiglio Nazionale delle Ricerche, Italy

Funnelling (Fun) is a recently proposed method for cross-lingual text classification (CLTC) based on a two-tier learning ensemble for heterogeneous transfer learning (HTL). In this ensemble method, 1st-tier classifiers, each working on a different and 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 meta-classifier that uses this vector as its input. The meta-classifier can thus exploit class-class correlations, and this (among other things) gives Fun an edge over CLTC systems in which these correlations cannot be brought to bear. In this paper we describe *Generalized Funnelling* (gFun), a generalisation of Fun consisting of an HTL architecture in which 1st-tier components can be arbitrary *view-generating functions*, i.e., language-dependent functions that each produce a language-independent representation ("view") of the (monolingual) document. We describe an instance of gFun in which the meta-classifier receives as input a vector of calibrated posterior probabilities (as in Fun) aggregated to other embedded representations that embody other types of correlations, such as word-class correlations (as encoded by *Word-Class Embeddings*), word-word correlations (as encoded by *Multilingual Unsupervised or Supervised Embeddings*), and word-context correlations (as encoded by *multilingual BERT*). We show that this instance of gFun substantially improves over Fun and over state-of-the-art baselines, by reporting experimental results obtained on two large, standard datasets for multilingual multilabel text classification. Our code that implements gFun is publicly available.

CCS Concepts: • **Computing methodologies → Ensemble methods**; *Supervised learning by classification.*

Additional Key Words and Phrases: Transfer Learning, Heterogeneous Transfer Learning, Cross-Lingual Text Classification, Ensemble Learning, Word Embeddings

1 INTRODUCTION

Transfer Learning (TL) [62] is a class of machine learning tasks in which, given a training set of labelled data items sampled from one or more "source" domains, we must issue predictions for unlabelled data items belonging to one or more "target" domains, related to the source domains but different from them. In other words, the goal of TL is to "transfer" (i.e., reuse) the knowledge that has been obtained from the training data in the source domains, to the target domains of interest, for which few labelled data (or no labelled data at all) exist. The rationale of TL is thus to increase the performance of a system on a downstream task (when few labelled data for this task exist),

*Also with Università di Pisa, Dipartimento di Informatica.

The order in which the authors are listed is purely alphabetical; each author has given an equally important contribution to this work.
Authors' address: Alejandro Moreo, alejandro.moreo@isti.cnr.it; Andrea Pedrotti, andrea.pedrotti@phd.unipi.it; Fabrizio Sebastiani, fabrizio.sebastiani@isti.cnr.it, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Via Giuseppe Moruzzi, 1, 56124, Pisa, Italy.

1046-8188/2022/6-ART $15.00
https://doi.org/10.1145/3544104

or to make it possible to carry out this task at all (when no training data at all for this task exist), while avoiding the cost of annotating new data items specific to this task.

TL techniques can be grouped into two main categories, according to the characteristics of the feature spaces in which the instances are represented. *Homogeneous* TL (which is often referred to as *domain adaptation* [69]) encompasses problems in which the source instances and the target instances are represented in a shared feature space. Conversely, *heterogeneous* TL [13] denotes the case in which the source data items and the target data items lie in different, generally non-overlapping feature spaces. This article focuses on the heterogeneous case only; from now on, by HTL we will thus denote *heterogeneous* transfer learning.

A prominent instance of HTL in the natural language processing and text mining areas is *Cross-Lingual Transfer Learning* (CLTL), in which data items have a textual nature and the different domains are actually different languages in which the data items are expressed. In turn, an important instance of CLTL is the task of *cross-lingual text classification* (CLTC), which consists of classifying documents, each written in one of a finite set $\mathcal{L} = \{\lambda_1, ..., \lambda_{|\mathcal{L}|}\}$ of languages, according to a shared *codeframe* (a.k.a. *classification scheme*) $\mathcal{Y} = \{y_1, ..., y_{|\mathcal{Y}|}\}$. The brand of CLTC we will consider in this paper is (cross-lingual) *multilabel* classification, namely, the case in which any document can belong to zero, one, or several classes at the same time.

The CLTC literature has focused on two main variants of this task. The first variant (that is sometimes called the *many-shot* variant) deals with the situation in which the target languages are such that language-specific training data are available for them as well; in this case, the goal of CLTC is to improve the performance of target language classification with respect to what could be obtained by leveraging the language-specific training data alone. If these latter data are few, the task if often referred to as *few-shot* learning. (We will deal with the many-shot/few-shot scenario in the experiments of Section 4.4.) The second variant is usually called the *zero-shot* variant, and deals with the situation in which there are no training data at all for the target languages; in this case, the goal of CLTC is to allow the generation of a classifier for the target languages, which could not be obtained otherwise. (We will deal with the zero-shot scenario in the experiments of Section 4.6.)

Many-shot CLTC is important, since in many multinational organisations (e.g., Vodafone, FAO, the European Union) many labelled data may be available in several languages, and there may be a legitimate desire to improve on the classification accuracy that monolingual classifiers are capable of delivering. The importance of few-shot and zero-shot CLTC instead lies in the fact that, while modern learning-based techniques for NLP and text mining have shown impressive performance when trained on huge amounts of data, there are many languages for which data are scarce. According to [29], the amount of (labelled and unlabelled) resources for the more than 7,000 languages spoken around the world follows (somehow unsurprisingly) a power-law distribution, i.e., while a small set of languages account for most of the available data, a very long tail of languages suffer from data scarcity, despite the fact that languages belonging to this long tail may have large speaker bases. Few-shot / zero-shot CLTL thus represents an appealing solution to dealing with this situation, since it attempts to bridge the gap between the high-resource languages and the low-resource ones.

However, the application of CLTC is not necessarily limited to scenarios in which the set of the source languages and the set of the target languages are disjoint, nor it is necessarily limited to cases in which there are few or no training data for the target domains. CLTC can also be deployed in scenarios where a language can play both the part of a source language (i.e., contribute to performing the task in other languages) and of a target language (i.e., benefit from training data expressed in other languages), and where sizeable quantities of labelled data exist for all languages at once. Such application scenarios, despite having attracted less research attention than the few-shot and zero-shot counterparts, are frequent in the context of multinational organisations, such as the European Union or UNESCO, or multilingual countries, such as India, South Africa, Singapore, and Canada, or multinational companies (e.g., Amazon, Vodafone). The aim of CLTC, in these latter cases, is to effectively exploit the potential synergies among the different languages in order to allow all languages to contribute to, and to benefit from, each other. Put it another way, the *raison d'être* of CLTC here becomes to deploy classification systems that perform substantially better than the trivial solution (the so-called *naïve classifier*) consisting of $|\mathcal{L}|$ monolingual classifiers trained independently of each other.

1.1 Funnelling and Generalized Funnelling

Esuli et al. [20] recently proposed *Funnelling* (Fun), an HTL method based on a two-tier classifier ensemble, and applied it to CLTC. In Fun, the 1st-tier of the ensemble is composed of $|\mathcal{L}|$ language-specific classifiers, one for each language in $\mathcal{L}$. For each document d, one of these classifiers (the one specific to the language of document d) returns a vector of $|\mathcal{Y}|$ calibrated posterior probabilities, where $\mathcal{Y}$ is the codeframe. Each such vector, irrespective of which among the $\mathcal{L}$ classifiers has generated it, is then fed to a 2nd-tier "meta-classifier" which returns the final label predictions.

The $|\mathcal{Y}|$-dimensional vector space to which the vectors of posterior probabilities belong, thus forms an "interlingua" among the $|\mathcal{L}|$ languages, since all these vectors are homologous, independently of which among the $|\mathcal{L}|$ classifiers have generated them. Another way of saying it is that all vectors are *aligned across languages*, i.e., the i-th dimension of the vector space has the same meaning in every language (namely, the "posterior" probability that the document belongs to class y_i). During training, the meta-classifier can thus learn from all labelled documents, irrespectively of their language. Given that the meta-classifier's prediction for each class in $\mathcal{Y}$ depends on the posterior probabilities received in input for all classes in $\mathcal{Y}$, the meta-classifier can exploit class-class correlations, and this (among other things) gives Fun an edge over CLTC systems in which these correlations cannot be brought to bear.

Fun was originally conceived with the many-shot / few-shot setting in mind; in such a setting, Fun proved superior to the naïve classifier and to 6 state-of-the-art baselines [20]. Esuli et al. [20] also sketched some architectural modifications that allow Fun to be applied to the zero-shot setting too.

In this paper we describe *Generalized Funnelling* (gFun), a generalisation of Fun consisting of an HTL architecture in which 1st-tier components can be arbitrary *view-generating functions* (VGFs), i.e., language-dependent functions that each produce a language-independent representation ("view") of the (monolingual) document. We describe an instantiation of gFun in which the meta-classifier receives as input, for the same (monolingual) document, a vector of calibrated posterior probabilities (as in Fun) as well as other language-independent vectorial representations, consisting of different types of document embeddings. These additional vectors are aggregated (e.g., via concatenation) with the original vectors of posterior probabilities, and the result is a set of extended, language-aligned, heterogeneous vectors, one for each monolingual document.

The original Fun architecture is thus a particular instance of gFun, in which the 1st-tier is equipped with only one VGF. The additional VGFs that characterize gFun each enable the meta-classifier to gain access to information on types of correlation in the data additional to the class-class correlations captured by the meta-classifier. In particular, we investigate the impact of *word-class correlations* (as embodied in *Word-Class Embeddings* (WCEs) [44]), *word-word correlations* (as embodied in *Multilingual Unsupervised or Supervised Embeddings* (MUSEs) [11]), and *correlations between contextualized words* (as embodied in embeddings generated by *multilingual BERT* [16]). As we will show, gFun natively caters for both the many-shot/few-shot and the zero-shot settings; we carry out extensive CLTC experiments in order to assess the performance of gFun in both cases. The results of these experiments show that mining additional types of correlations in data does make a difference, and that gFun outperforms Fun as well as other CLTC systems that have recently been proposed.

The rest of this article is structured as follows. In Section 2 we describe the gFun framework, while in Section 3 we formalize the concept of "view-generating function" and present several instances of it. Section 4 reports the experiments (for both the many-shot and the zero-shot variants)[1] that we have performed on two large datasets for multilingual multilabel text classification. In Section 5 we move further and discuss a more advanced,

[1]We do not explicitly present experiments for the few-shot case since a few-shot system is technically no different from a many-shot system.

"recurrent" VGF that combines MUSEs and WCEs in a more sophisticated way, and test it in additional experiments. We review related work and methods in Section 6. In Section 7 we conclude by sketching avenues for further research. Our code that implements gFun is publicly available.[2]

2 GENERALIZED FUNNELLING

In this section, we first briefly summarise the original Fun method, and then move on to present gFun and related concepts.

2.1 A brief introduction to Funnelling

Funnelling, as described in [20], comes in two variants, called Fun(tat) and Fun(kfcv). We here disregard Fun(kfcv) and only use Fun(tat), since in all the experiments reported in [20] Fun(tat) clearly outperformed Fun(kfcv); see [20] if interested in a description of Fun(kfcv). For ease of notation, we will simply use Fun to refer to Fun(tat).

In Fun (see Figure 1), in order to train a classifier ensemble, 1st-tier language-specific classifiers $h_1^1, ..., h_{|\mathcal{L}|}^1$ (with superscript 1 indicating the 1st tier) are trained from their corresponding language-specific training sets $\text{Tr}_1, ..., \text{Tr}_{|\mathcal{L}|}$. Training documents $d \in \text{Tr}_i$ may be represented by means of any desired vectorial representation $\phi_i^1(d) = \mathbf{d}$, such as, e.g., TFIDF-weighted bag-of-words, or character n-grams; in principle, different styles of vectorial representation can be used for the different 1st-tier classifiers, if desired. The classifiers may be trained by any learner, provided the resulting classifier returns, for each language λ_i, document d, and class y_j, a confidence score $h_i^1(\mathbf{d}, y_j) \in \mathbb{R}$; in principle, different learners can be used for the different 1st-tier classifiers, if desired.

Each 1st-tier classifier h_i^1 is then applied to each training document $d \in \text{Tr}_i$, thus generating a vector

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

of confidence scores for each $d \in \text{Tr}_i$. (Incidentally, this is the phase in which Fun(tat) and Fun(kfcv) differ, since Fun(kfcv) uses instead a k-fold cross-validation process to classify the training documents.)

The next step consists of computing (via a chosen probability calibration method) language- and class-specific calibration functions f_{ij} that map confidence scores $h_i^1(\mathbf{d}, y_j)$ into calibrated posterior probabilities $\Pr(y_j|\mathbf{d})$.[3]

Fun then applies f_{ij} to each confidence score and obtains a vector of calibrated posterior probabilities

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

Note that the i index for language λ_i has disappeared, since calibrated posterior probabilities are comparable across different classifiers, which means that we can use a shared, language-independent space of vectors of calibrated posterior probabilities.

At this point, the 2nd-tier, language-independent "meta"-classifier h^2 can be trained from all training documents $d \in \bigcup_{i=1}^{|\mathcal{L}|} \text{Tr}_i$, 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 $d \in \text{Te}_i$ from language λ_i, Fun applies classifier h_i^1 to $\phi_i^1(d) = \mathbf{d}$ and converts the resulting vector $S(d)$ of confidence scores into a vector $\phi^2(d)$ of calibrated posterior probabilities. Fun then feeds this latter to the meta-classifier h^2, which returns (in the case of multilabel classification) a vector of binary labels representing the predictions of the meta-classifier.

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

[3]The reason why we need calibration is that the confidence scores obtained from different classifiers are not comparable; the calibration process serves the purpose of mapping these confidence scores into entities (the calibrated posterior probabilities) that are indeed comparable even if originating from different classifiers.

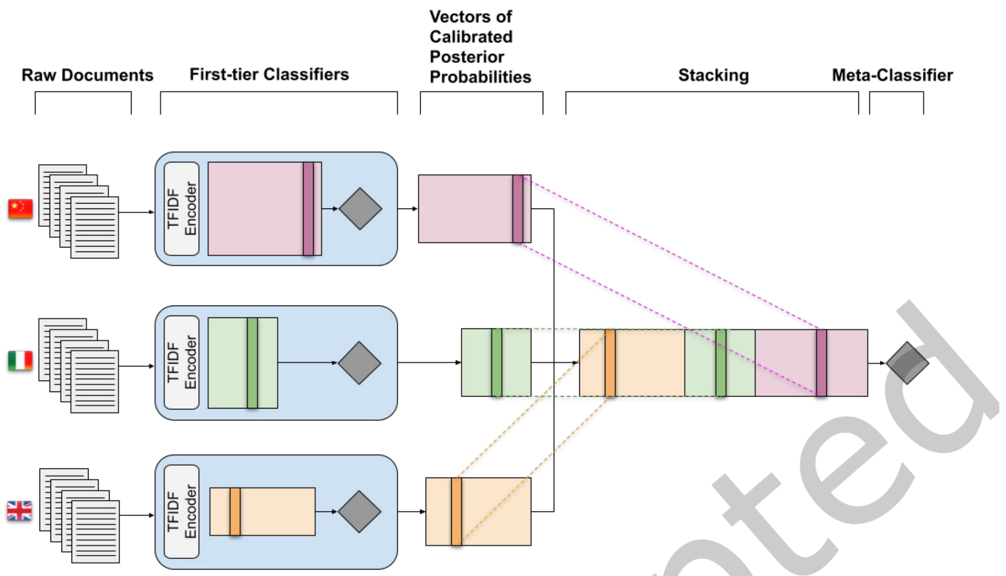

Fig. 1. The FUN architecture, exemplified with $|\mathcal{L}|$=3 languages (Chinese, Italian, English). Note that the different term-document matrices in the 1st-tier may contain different numbers of documents and/or different numbers of terms. The three grey diamonds on the left represent calibrated classifiers that map the original vectors (e.g., TFIDF vectors) into $|\mathcal{Y}|$-dimensional spaces. The resulting vectors are thus aligned and can all be used for training the meta-classifier, which is represented by the grey diamond on the right.

2.2 Introducing heterogeneous correlations through Generalized Funnelling

As explained in [20], the reasons why FUN outperforms the naïve monolingual baseline consisting of $|\mathcal{L}|$ independently trained, language-specific classifiers, are essentially two. The first is that FUN learns from heterogeneous data; i.e., while in the naïve monolingual baseline each classifier is trained only on $|\text{Tr}_i|$ labelled examples, the meta-classifier in FUN is trained on all the $\bigcup_{i=1}^{|\mathcal{L}|} |\text{Tr}_i|$ labelled examples. Put it another way, in FUN all training examples contribute to classifying all unlabelled examples, irrespective of the languages of the former and of the latter. The second is that the meta-classifier leverages *class-class correlations*, i.e., it learns to exploit the stochastic dependencies between classes typical of multiclass settings. In fact, for an unlabelled document d the meta-classifier receives $|\mathcal{Y}|$ inputs from the 1st-tier classifier which has classified d, and returns $|\mathcal{Y}|$ confidence scores, which means that the input for class y' has a potential impact on the output for class y'', for every y' and y''.

In FUN, the key step in allowing the meta-classifier to leverage the different language-specific training sets consists of mapping all the documents onto a space shared among all languages. This is made possible by the fact that the 1st-tier classifiers all return vectors of calibrated posterior probabilities. These vectors are homologous (since the codeframe is the same for all languages), and are also comparable (because the posterior probabilities are calibrated), which means that we can have all vectors share the same vector space irrespectively of the language of provenance.

In gFUN, we generalize this mapping by allowing a set Ψ of *view-generating functions* (VGFs) to define this shared vector space. VGFs are language-dependent functions that map (monolingual) documents into language-independent vectorial representations (that we here call *views*) aligned across languages. Since each view is aligned

```
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
           $\mathcal{Y} = \{y_1, ..., y_{|\mathcal{Y}|}\}$;
         • Set $\Psi = \{\psi_1, ..., \psi_{|\Psi|}\}$ of VGFs;
Output : • VGF parameters $\Theta = \{\theta_{ik}\}, 1 \leq i \leq |\mathcal{L}|, 1 \leq k \leq |\Psi|$ ;
         • Parameters of the aggregation function $\Lambda$
         • Meta-classifer $h^2$
1  for $\psi_k \in \Psi$ do
2      /* Learn the parameters of the $k$th VGF for each language $\lambda_i$ */
3      for $\lambda_i \in \mathcal{L}$ do
4          $\theta_{ik} \leftarrow fit(\psi_k, \mathrm{Tr}_i)$ ;
5      end
6      /* Stack all language views produced by $\psi_k$ */
7      $\mathrm{V}_k \leftarrow \mathrm{vstack}(\psi_k(\mathrm{Tr}_1, \theta_{1k}), \ldots, \psi_k(\mathrm{Tr}_{|\mathcal{L}|}, \theta_{|\mathcal{L}|k}))$ ;
8  end
9  /* Learn the parameters (if any) of the aggregation function */
10 $\Lambda \leftarrow fit(aggfunc, \ldots)$ ;
11 /* Combine all training sets by aggregating the language-independent views */
12 $\mathrm{Tr}' \leftarrow aggfunc(V_1, \ldots, V_{|\Psi|}, \Lambda)$ ;
13 Train meta-classifier $h^2$ from all vectors in $\mathrm{Tr}'$ ;
14 $\Theta \leftarrow \{\theta_{ik}\}, 1 \leq i \leq |\mathcal{L}|, 1 \leq k \leq |\Psi|$ ;
15 return $\Lambda$, $\Theta$, $h^2$
```

Algorithm 1: Generalized Funnelling for CLTC, training phase.

across languages, it is easy to aggregate (e.g., by concatenation) the different views of the same monolingual document into a single representation that is also aligned across languages, and which can be thus fed to the meta-classifier.

Different VGFs are meant to encode different types of information so that they can all be brought to bear on the training process. In the present paper we will experiment with extending Fun by allowing views consisting of different types of document embeddings, each capturing a different type of correlation within the data.

The procedures for training and testing cross-lingual classifiers via gFun are described in Algorithm 1 and Algorithm 2, respectively. The first step of the training phase is the optimisation of the parameters (if any) of the VGFs $\psi_k \in \Psi$ (Algorithm 1 – Line 4), which is carried out independently for each language and for each VGF. A VGF ψ_k produces representations that are aligned across all languages, which means that vectors coming from different languages can be "stacked" (i.e., placed in the same set) to define the view V_k (Algorithm 1 – Line 7), which corresponds to the ψ_k portion of the entire (now language-independent) training set of the meta-classifier. Note that the vectors in a given view need not be probabilities; we only assume that they are homologous and comparable across languages. The aggregation function (*aggfunc*) implements a policy for aggregating the different views for them to be input to the meta-classifier; it is thus used both during training (Algorithm 1 – Line 12) and during test (Algorithm 2 – Line 3). In case the aggregation function needs to learn some parameters, those are estimated during training (Algorithm 1 – Line 10).

Finally, note that both the training phase and the test phase are highly parallelisable, since the (training and/or testing) data for language λ' can be processed independently of the analogous data for language λ'', and since each view within a given language can be generated independently of the other views for the same language.

Note that the original formulation of Fun (Section 2.1) thus reduces to an instance of gFun in which there is a single VGF (one that converts documents into calibrated posterior probabilities) and the aggregation function is simply the identity function. In this case, the fit of the VGF (Algorithm 1 – Line 4) comes down to computing

```
Input  : • Sets $\{\text{Te}_1, ..., \text{Te}_{|\mathcal{L}|}\}$ 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 VGFs with parameters $\Theta = \{\theta_{ik}\}, 1 \leq i \leq |\mathcal{L}|, 1 \leq k \leq |\Psi|$ ;
         • Parameters $\Lambda$ of the aggregation function ;
         • meta-classifier $h^2$ ;
Output : • Labels for all documents in $\{\text{Te}_1, ..., \text{Te}_{|\mathcal{L}|}\}$ ;
1 for $\lambda_i \in \mathcal{L}$ do
2     /* Aggregate the views produced by all VGFs */
3     $\text{Te}'_i \leftarrow aggfunc(\psi_1(\text{Te}_i, \theta_{i1}), \ldots, \psi_{|\Psi|}(\text{Te}_i, \theta_{i|\Psi|}), \Lambda)$ ;
4     /* Use the meta-classifier $h^2$ to predict labels $L_i$ for all documents in $\text{Te}'_i$ */
5     $L_i \leftarrow h^2(\text{Te}'_i)$
6 end
7 return $\{L_1, \ldots, L_{|\mathcal{L}|}\}$
```

Algorithm 2: Generalized Funnelling for CLTC, testing phase.

weighted (e.g., via TFIDF) vectorial representations of the training documents, training the 1st-tier classifiers, and calibrating them. Examples of the parameters obtained as a result of the fitting process include the choice of vocabulary, the IDF scores, the parameters of the separating hyperplane, and those of the calibration function. During the test phase, invoking the VGF (Algorithm 2 – Line 3) amounts to computing the weighted vectorial representations and the $\phi^2(d)$ representations (Equation 2) of the test documents, using the classifiers and meta-classifier generated during the training phase.

In what follows we describe the VGFs that we have investigated in order to introduce into gFun sources of information additional to the ones that are used in Fun. In particular, we describe in detail each such VGF in Sections 3.1-3.4, we discuss aggregation policies in Section 3.5, and we analyse a few additional modifications concerning data normalisation (Section 3.6) that we have introduced into gFun and that, although subtle, bring about a substantial improvement in the effectiveness of the method.

3 VIEW-GENERATING FUNCTIONS

In this section we describe the VGFs that we have investigated throughout this research, by also briefly explaining related concepts and works from which they stem.

As already stated, the main idea behind our instantiation of gFun is to learn from heterogeneous information about different kinds of correlations in the data. While the main ingredients of the text classification task are words, documents, and classes, the key to approach the CLTC setting lies in the ability to model them consistently across all languages. We envision ways for bringing to bear the following stochastic correlations among these elements:

(1) Correlations between different classes: understanding how classes are related to each other in some languages may bring about additional knowledge useful for classifying documents in other languages. These correlations are specific to the particular codeframe used, and are obviously present only in multilabel scenarios. They can be used (in our case: by the meta-classifier) in order to refine an initial classification (in our case: by the 1st-tier classifiers), since they are based on the relationships between posterior probabilities / labels assigned to documents.
(2) Correlations between different words: by virtue of the "distributional hypothesis" (see [52]), words are often modelled in accordance to how they are distributed in corpora of text with respect to other words. Distributed representations of words encode the relationships between words and other words; when properly aligned across languages, they represent an important help for bringing lexical semantics to bear

on multilingual text analysis processes, thus helping to bridge the gap among language-specific sources of labelled information.

(3) Correlations between words and classes: profiling words in terms of how they are distributed across the classes in a language is a direct way of devising cross-lingual word embeddings (since translation-equivalent words are expected to exhibit similar class-conditional distributions), which is compliant with the distributional hypothesis (since semantically similar words are expected to be distributed similarly across classes).

(4) Correlations between contextualized words: the meaning of a word occurrence is dependent on the specific context in which the word occurrence is found. Current language models are well aware of this fact, and try to generate contextualized representations of words, which can in turn be used straightforwardly in order to obtain contextualized representations for entire documents. Language models trained on multilingual data are known to produce distributed representations that are coherent across the languages they have been trained on.

We recall from Section 2.1 that class-class correlations are exploited in the 2nd-tier of Fun. We model the other types of correlations mentioned above via dedicated VGFs. We investigate instantiations of the aforementioned correlations by means of independently motivated modular VGFs. Here we provide a brief overview of each them.

- *the Posteriors VGF*: it maps documents into the space defined by calibrated posterior probabilities. This is, aside from the modifications discussed in Section 3.6, equivalent to the 1st-tier of the original Fun, but we discuss it in detail in Section 3.1.
- *the MUSEs VGF* (encoding correlations between different words): it uses the (supervised version of) Multi-lingual Unsupervised or Supervised Embeddings (MUSEs) made available by the authors of [11]. MUSEs have been trained on Wikipedia[4] in 30 languages and have later been aligned using bilingual dictionaries and iterative Procrustes alignment (see Section 3.2 and [11]).
- *the WCEs VGF* (encoding correlations between words and classes): it uses Word-Class Embeddings (WCEs) [44], a form of supervised word embeddings based on the class-conditional distributions observed in the training set (see Section 3.3).
- *the BERT VGF* (encoding correlations between different contextualized words): it uses the contextualized word embeddings generated by multilingual BERT [17], a deep pretrained language model based on the transformer architecture (see Section 3.4).

In the following sections we present each VGF in detail.

3.1 The Posteriors VGF

This VGF coincides with the 1st-tier of Fun, but we briefly explain it here for the sake of completeness.

Here the idea is to leverage the fact that the classification scheme is common to all languages, in order to define a vector space that is aligned across all languages. Documents, regardless of the language they are written in, can be redefined with respect to their relations to the classes in the codeframe. Using a geometric metaphor, the relation between a document and a class can be defined in terms of the distance between the document and the surface that separates the class from its complement. In other words, while the language-specific vector spaces where the original document vectors lie are not aligned (e.g., they can be characterized by different numbers of dimensions, and the dimensions for one language bear no relations to the dimensions for another language), one can profile each document via a new vector consisting of the distances to the separating surfaces relative to the various classes. By using the binary classifiers as "pivots" [1], documents end up being represented in a shared space, in which the number of dimensions are the same for all languages (since the classes are assumed to be

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

the same for all languages), and the vector values for each dimension are comparable across languages once the distances to the classification surfaces are properly normalized (which is achieved by the calibration process).

Note that this procedure is, in principle, independent of the characteristics of any particular vector space and learning device used across languages, both of which can be different across the languages.[5]

For ease of comparability with the results reported by Esuli et al. [20], in this paper we will follow these authors and encode (for all languages in $\mathcal{L}$) documents as bag-of-words vectors weighted via TFIDF, which is computed as

$$\text{TFIDF}(w_k, \mathbf{x}_j) = \text{TF}(w_k, \mathbf{x}_j) \cdot \log \frac{|\text{Tr}|}{\#_{\text{Tr}}(w_k)} \tag{3}$$

where $\#_{\text{Tr}}(w_k)$ is the number of documents in Tr in which word w_k occurs at least once and

$$\text{TF}(w_k, \mathbf{x}_j) = \begin{cases} 1 + \log \#(w_k, \mathbf{x}_j) & \text{if } \#(w_k, \mathbf{x}_j) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\#(w_k, \mathbf{x}_j)$ stands for the number of times w_k appears in document $\mathbf{x}_j$. Weights are then normalized via cosine normalisation, as

$$w(w_k, \mathbf{x}_j) = \frac{\text{TFIDF}(w_k, \mathbf{x}_j)}{\sqrt{\sum_{w' \in d_j} \text{TFIDF}(w'_k, \mathbf{x}_j)^2}} \tag{5}$$

For the very same reasons we also follow [20] in adopting (for all languages in $\mathcal{L}$) Support Vector Machines (SVMs) as the learning algorithm, and "Platt calibration" [50] as the probability calibration function.

3.2 The MUSEs VGF

In CLTL, the need to transfer lexical knowledge across languages has given rise to cross-lingual representations of words in a joint space of embeddings. In our research, in order to encode word-word correlations across different languages we derive document embeddings from (the supervised version of) *Multilingual Unsupervised or Supervised Embeddings* (MUSEs) [11]. MUSEs are word embeddings generated via a method for aligning unsupervised (originally monolingual) word embeddings in a shared vector space, similar to the method described in [39]. The alignment is obtained via a linear mapping (i.e., a rotation matrix) 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, i.e., 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. The mapping is then further refined using a technique called "Procrustes alignment". The qualification "Unsupervised or Supervised" refers to the fact that the method can operate with or without a dictionary of parallel seed words; we use the embeddings generated in supervised fashion.

We use the MUSEs that Conneau et al. [11] make publicly available[6], and that consist of 300-dimensional multilingual word embeddings trained on Wikipedia using fastText. To date, the embeddings have been aligned for 30 languages with the aid of bilingual dictionaries.

Fitting the VGF for MUSEs consists of first allocating in memory the pre-trained MUSE matrices $\mathbf{M}_i \in \mathbb{R}^{(v_i \times 300)}$ (where v_i is the vocabulary size for the i-th language), made available by Conneau et al. [11], for each language

[5]The vector spaces can indeed be completely different from one language to another. For example, one could define a bag of TFIDF-weighted bigrams for English, a bag of BM25-weighted unigrams for French, and an SVD-decomposed space for Spanish. Note also that the learning algorithms can be different as well; one may use, say, SVMs for English, logistic regression for French, and AdaBoost for Spanish. As long as the decision scores provided by each classifier are turned into calibrated posterior probabilities, the language-specific representations can be recast into language-independent, comparable representations.
[6]https://github.com/facebookresearch/MUSE

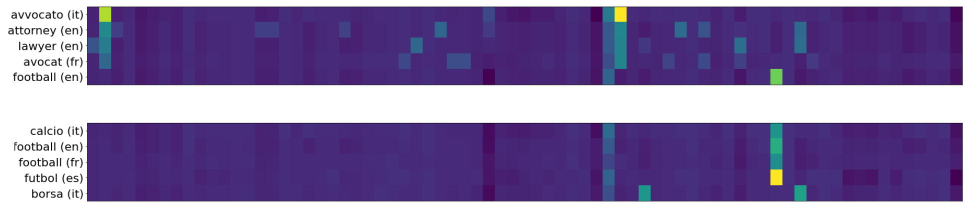

Fig. 2. Heatmaps displaying five WCEs each, where each cell indicates the correlation between a word (row) and a class (column), as from the RCV1/RCV2 dataset. Yellow indicates a high value of correlation while blue indicates a low such value. Words “avvocato” and “avocat” are Italian and French translations, resp., of the English word “lawyer”; words “calcio” and “futbol” are Italian and Spanish translations, resp., of the English word “football”; Italian word “borsa” instead means “bag”.

λ_i involved, and then generating document embeddings for all training documents as weighted averages of the words in the document. As the weighting function, we use TFIDF (Equation 3). This computation reduces to performing the projection $\mathbf{X}_i \cdot \mathbf{M}_i$, where the matrix $\mathbf{X}_i \in \mathbb{R}^{(|\mathrm{Tr}_i| \times v_i)}$ consists of the TFIDF-weighted vectors that represent the training documents (for this we can reuse the matrices $\mathbf{X}_i$ computed by the Posteriors VGF, since they are identical to the ones needed here). The process of generating the views of test documents via this VGF is also obtained via a projection $\mathbf{X}_i \cdot \mathbf{M}_i$, where in this case the $\mathbf{X}_i$ matrix consists of the TFIDF-weighted vectors that represent the *test* documents.

3.3 The WCEs VGF

In order to encode word-class correlations we derive document embeddings from *Word-Class Embeddings* (WCEs [44]). WCEs are supervised embeddings meant to extend (e.g., by concatenation) other unsupervised pre-trained word embeddings (e.g., those produced by means of `word2vec`, or `GloVe`, or any other technique) in order to inject task-specific word meaning in multiclass text classification. The WCE for word w is defined as

$$E(w) = \varphi(\eta(w, y_1), \ldots, \eta(w, y_{|\mathcal{Y}|})) \tag{6}$$

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 [44], whose computation is very efficient; as φ we use the identity function, which means that our WCEs are $|\mathcal{Y}|$-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 languages $\lambda_i \in \mathcal{L}$. Document embeddings relying on WCEs thus display similar characteristics irrespective of the language in which the document is written in. In fact, given a set of documents classified according to a common codeframe, WCEs reflect the intuition that words that are semantically similar across languages (i.e., are translations of each other) tend to exhibit similar correlations to the classes in the codeframe. This is, to the best of our knowledge, the first application of WCEs to a multilingual setting.

The intuition behind this idea is illustrated by the two examples in Figure 2, where two heatmaps display the correlation values of five WCEs each. Each of the two heatmaps illustrates the distribution patterns of four terms that are either semantically related or translation equivalents of each other (first four rows in each subfigure), and of a fifth term semantically unrelated to the previous four (last row in each subfigure). Note that not only semantically related terms in a language get similar representations (as is the case of “attorney” and “lawyer” in English), but also translation-equivalent terms do so (e.g., “avvocato” in Italian and “avocat” in French).

The VGF for WCEs is similar to that for MUSEs, but for the fact that in this case the matrix containing the word embeddings needs to be obtained from our training data, and is not pretrained on external data. More specifically, fitting the VGF for WCEs comes down to first computing, for each language $\lambda_i \in \mathcal{L}$, the language-specific WCE matrix $\mathbf{W}_i$ according to the process outlined in [44], and then projecting the TFIDF-weighted matrix $\mathbf{X}_i$ obtained from Tr_i, as $\mathbf{X}_i \cdot \mathbf{W}_i$. (Here too, we use the TFIDF variant of Equation 3.) During the testing phase, we simply perform the same projection $\mathbf{X}_i \cdot \mathbf{W}_i$ as above, where $\mathbf{X}_i$ now represents the weighted matrix obtained from the test set.

Although alternative ways of exploiting word-class correlations have been proposed in the literature, we adopted WCEs because of their higher simplicity with respect to other methods. For example, the GILE system [46] uses label descriptions in order to compute a model of compatibility between a document embedding and a label embedding; differently from the latter work, in our problem setting we do not assume to have access to textual descriptions of the semantics of the labels. The LEAM model [64], instead, defines a word-class attention mechanism and can work with or without label descriptions (though the former mode is considered preferable), but has never been tested in multilingual contexts; preliminary experiments we have carried out by replacing the GloVe embeddings originally used in LEAM with MUSE embeddings, have not produced competitive results.

3.4 The BERT VGF

BERT [17] is a bidirectional language model based on the transformer architecture [61] trained on a *masked language modelling* objective and *next sentence prediction* task. The transformer architecture has been initially proposed for the task of sequence transduction relying solely on the attention mechanism, and thus discarding the usual recurrent components deployed in encoder-decoder architectures. BERT's transformer blocks contain two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. Differently from other architectures [49], BERT's attention is set to attend to all the input tokens (i.e., it deploys bidirectional self-attention), thus making it well-suited for sentence-level tasks. Originally, the BERT architecture was trained by Devlin et al. [17] on a monolingual corpus composed of the BookCorpus and English Wikipedia (for a total of roughly 3,300M words). Recently, a multilingual version, called mBERT [16], has been released. The model is no different from the standard BERT model; however, mBERT has been trained on concatenated documents gathered from Wikipedia in 104 different languages. Its multilingual capabilities emerge from the exposure to different languages during this massive training phase.

In this research, we explore mBERT as a VGF for gFun. At training time, this VGF is first equipped with a fully-connected output layer, so that BERT can be trained end-to-end using binary cross-entropy as the loss function. Nevertheless, as its output (i.e., the one that is eventually fed to the meta-classifier also at testing time) we use the hidden state associated with the document embedding (i.e., the `[CLS] token`) at its last layer.

3.5 Policies for aggregating VGFs

The different views of the same document that are independently generated by the different VGFs need to be somehow merged together before being fed to the meta-classifier. This is undertaken by operators that we call *aggregation functions*. We explore two different policies for view aggregation: concatenation and averaging.

Concatenation simply consists of juxtaposing, for a given document, the different views of this document, thus resulting in a vector whose dimensionality is the sum of the dimensionalities of the contributing views. This policy is the more straightforward one, and one that does not impose any constraint on the dimensionality of the individual views as generated from different VGFs.

Averaging consists instead of computing, for a given document, a vector which is the average of the different views for this document. In order for it to be possible, though, this policy requires that the views (i) all have the same dimensionality, and (ii) are aligned among each other, i.e., that the i-th dimension of the vector has the same meaning in every view. This is obviously not the case with the views produced by the VGFs we have described up to now. In order to solve this problem, we learn additional mappings onto the space of class-conditional posterior probabilities, i.e., for each VGF (other than the Posteriors VGF of Section 3.1, which already returns vectors of $|\mathcal{Y}|$ calibrated posterior probabilities) we train a classifier that maps the view of a document into a vector of $|\mathcal{Y}|$ calibrated posterior probabilities. The net result is that each document d is represented by m vectors of $|\mathcal{Y}|$ calibrated posterior probabilities (where m is the number of VGFs in our system). These m vectors can be averaged, and the resulting average vector can be fed to the meta-classifier as the only representation of document d. The way we learn the above mappings is the same used in Fun; this also brings about uniformity between the vectors of posterior probabilities generated by the Posteriors VGF and the ones generated by the other VGFs. Note that in this case, though, the classifier for VGF ψ_k is trained on the views produced by ψ_k for *all* training documents, irrespectively of their language of provenance; in other words, for performing these mappings we just train $(m-1)$ (and not $(m-1) \times |\mathcal{L}|$) classifiers, one for each VGF other than the Posteriors VGF.

Each of these two aggregation policies has different pros and cons.

The main advantage of concatenation is that it is very simple to implement. However, it suffers from the fact that the number of dimensions in the resulting dense vector space is high, thus leading to a higher computational cost for the meta-classifier. Above all, since the number of dimensions that the different views contribute is not always the same, this space (and the decisions of the meta-classifier) can be eventually dominated by the VGFs characterized by the largest number of dimensions.

The averaging policy (Figure 3), on the other hand, scales well with the number of VGFs, but requires learning additional mappings aimed at homogenising the different views into a unified representation that allows averaging them. Despite the additional cost, the averaging policy has one appealing characteristic, i.e., *the 1st-tier is allowed to operate with different numbers of VGFs for different languages* (provided that there is at least one VGF per language); in fact, the meta-level representations are simply computed as the average of the views that are available for that particular language. For reasons that will become clear in Section 4.6, this property allows gFun to natively operate in zero-shot mode.

In Section 4.7 we briefly report on some preliminary experiments that we had carried out in order to assess the relative merits of the two aggregation policies in terms of classification performance. As we will see in Section 4.7 in more detail, the results of those experiments indicate that, while differences in performance are small, they tend to be in favour of the averaging policy. This fact, along with the fact that the averaging policy scales better with the number of VGFs, and along with the fact that it allows different numbers of VGFs for different languages, will eventually lead us to opt for averaging as our aggregation policy of choice.

3.6 Normalisation

We have found that applying some routine normalisation techniques to the vectors returned by our VGFs leads to consistent performance improvements. This normalisation consists of the following steps:

(1) Apply (only for the MUSEs VGF and WCEs VGF) *smooth inverse frequency* (SIF) [3] to remove the first principal component of the document embeddings obtained as the weighted average of word embeddings. In their work, Arora et al. [3] show that removing the first principal component from a matrix of document embeddings defined as a weighted average of word embeddings, is generally beneficial. The reason is that the way in which most word embeddings are trained tends to favour the accumulation of large components along semantically meaningless directions. However, note that for the MUSEs VGF and WCEs VGF we use

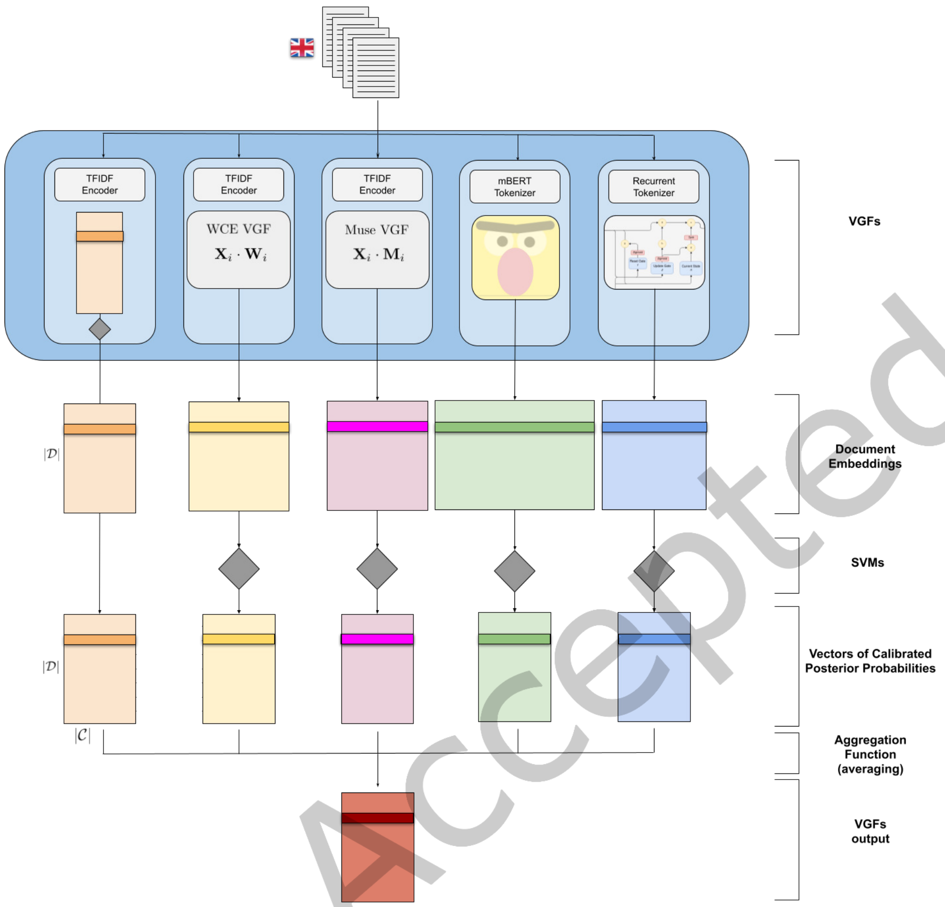
TFIDF Encoder
TFIDF Encoder
TFIDF Encoder
mBERT Tokenizer
Recurrent Tokenizer
WCE VGF $\mathbf{X}_i \cdot \mathbf{W}_i$
Muse VGF $\mathbf{X}_i \cdot \mathbf{M}_i$
VGFs
$|\mathcal{D}|$
Document Embeddings
SVMs
$|\mathcal{D}|$
Vectors of Calibrated Posterior Probabilities
$|\mathcal{C}|$
Aggregation Function (averaging)
VGFs output

Fig. 3. The averaging policy for view aggregation: the views are recast in terms of vectors of calibrated posterior probabilities before being averaged. Note that the resulting vectors lie in the same vector space. For ease of visualisation, only one language (English) is shown.

the TFIDF weighting criterion instead of the criterion proposed by Arora et al. [3], since in our case we are modelling (potentially large) documents, instead of sentences like in their case.[7]

(2) Impose unit L2-norm to the vectors before aggregating them by means of concatenation or averaging.

(3) Standardize[8] the columns of the language-independent representations before training the classifiers (this includes (a) the classifiers in charge of homogenising the vector spaces before applying the averaging policy, and (b) the meta-classifier).

[7]The weighting technique proposed by Arora et al. [3] does not account for term repetitions, since they make the assumption that words rarely occur more than once in a sentence. Conversely, when modelling entire documents, the TF factor may indeed play a fundamental role, and in such cases, as Arora et al. [3] acknowledge, using TFIDF may be preferable.

[8]Standardising (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.

The rationale behind these normalisation steps, when dealing with heterogeneous representations, is straightforward and two-fold. On one side, it is a means for equating the contributions brought to the model by the different sources of information. On the other, it is a way to counter the internal covariate shift across the different sources of information (similar intuitions are well-known and routinely applied when training deep neural architectures – see, e.g., [27]).

What might come as a surprise is the fact that normalisation helps improve gFun even when we equip gFun only with the Posteriors VGF (which coincides with the original Fun architecture), and that this improvement is statistically significant. We quantify this variation in performance in the experiments of Section 4.

4 EXPERIMENTS

In order to maximize the comparability with previous results we adopt an experimental setting identical to the one used in [20], which we briefly sketch in this section. We refer the reader to [20] for a more detailed discussion of this experimental setting.

4.1 Datasets

The first of our two datasets is a version (created by Esuli et al. [20]) of RCV1/RCV2, a corpus of news stories published by Reuters. This version of RCV1/RCV2 contains documents each written in one of 9 languages (English, Italian, Spanish, French, German, Swedish, Danish, Portuguese, and Dutch) and classified according to a set of 73 classes. The dataset consists of 10 random samples, obtained from the original RCV1/RCV2 corpus, each consisting of 1,000 training documents and 1,000 test documents for each of the 9 languages (Dutch being an exception, since only 1,794 Dutch documents are available; in this case, each sample consists of 1,000 training documents and 794 test documents). Note though that, while each random sample is balanced at the language level (same number of training documents per language and same number of test documents per language), it is not balanced at the class level: at this level the dataset RCV1/RCV2 is highly imbalanced (the number of documents per class ranges from 1 to 3,913 – see Table 1), and each of the 10 random samples is too. The fact that each language is equally represented in terms of both training and test data allows the many-shot experiments to be carried out in controlled experimental conditions, i.e., minimizes the possibility that the effects observed for the different languages are the result of different amounts of training data. (Of course, zero-shot experiments will instead be run by excluding the relevant training set(s).) Both the original RCV1/RCV2 corpus and the version we use here are comparable at topic level, as news stories are not direct translations of each other but simply discuss the same or related events in different languages.

The second of our two datasets is a version (created by Esuli et al. [20]) of JRC-Acquis, a corpus of legislative texts published by the European Union. This version of JRC-Acquis contains documents each written in one of 11 languages (the same 9 languages of RCV1/RCV2 plus Finnish and Hungarian) and classified according to a set of 300 classes. The dataset is parallel, i.e., each document is included in 11 translation-equivalent versions, one per language. Similarly to the case of RCV1/RCV2 above, the dataset consists of 10 random samples, obtained from the original JRC-Acquis corpus, each consisting of at least 1,000 training documents for each of the 11 languages (summing up to a total of 12,687 training documents in each sample), and 4,242 test documents for each of the 11 languages. As in the case of RCV1/RCV2, this version of JRC-Acquis is not balanced at the class level (the number of positive examples per class ranges from 55 to 1,155), and the samples obtained from it are not balanced either. Note that, in this case, Esuli et al. [20] included at most one of the 11 language-specific versions in a training set, in order to avoid the presence of translation-equivalent content in the training set; this enables one to measure the contribution of training information coming from different languages in a more realistic setting. When a document is included in a test set, instead, all its 11 language-specific versions are also included, in order to allow

	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{Y}\rvert$	$\lvert$Tr$\rvert$	$\lvert$Te$\rvert$	Ave.Cls	Min.Cls	Max.Cls	Min.Pos	Max.Pos	Ave.Feats
RCV1/RCV2	9	73	9,000	8,794	3.21	1	13	1	3,913	4,176
JRC-Acquis	11	300	12,687	46,662	3.31	1	18	55	1,155	9,909

Table 1. Characteristics of the datasets used in [20] and in this paper, including the number of languages ($|\mathcal{L}|$); number of classes ($|\mathcal{Y}|$); number of training ($|$Tr$|$) and test ($|$Te$|$) documents; average (Ave.Cls), minimum (Min.Cls), and maximum (Max.Cls) number of classes per document; minimum (Min.Pos) and maximum (Max.Pos) number of positive examples per class; and average number of distinct features per language (Ave.Feats).

Text	Labels
BRAZIL: Talks stall on bill to scrap Brazil export tax. Voting to speed up a bill to remove a tax on Brazilian exports will take place August 27 at the earliest after federal and state governments failed to reach an accord on terms, a Planning Ministry spokeswoman said. Planning Minister Antonio Kandir and the Parana and Rio Grande do Sul governments have yet to agree on compensation following the proposed elimination of the so-called ICMS tax, which applies to products such as coffee, sugar and soyproducts. The elimination of the tax should inject at least $1.5 billion into the agribusiness sector (...) [Other 505 words truncated]	• merchandise trade (E512) • economics (ECAT) • government finance (E21) • trade/reserves (E51) • expediture/revenue (E211)
Commission Regulation (EC) No 1908/2004 of 29 October 2004 fixing the maximum aid for cream, butter and concentrated butter for the 151th individual invitation to tender under the standing invitation to tender provided for in Regulation (EC) No 2571/97 THE COMMISSION OF THE EUROPEAN COMMUNITIES, Having regard to the Treaty establishing the European Community, Having regard to Council Regulation (EC) No 1255/1999 of 17 May 1999 on the common organisation of the market in milk and milk products [1], and in particular Article 10 thereof, Whereas: (1) The intervention agencies are, pursuant to Commission Regulation (EC) No 2571/97 of 15 December 1997 on the sale of butter (...) [Other 243 words truncated]	• award of contract (20) • concentrated product (2741) • aid system (3003) • farm price support (4236) • butter (4860) • youth movement (2004)

Table 2. Excerpts from example documents from RCV1/RCV2 (1st example) and JRC-Acquis (2nd example).

a perfectly fair evaluation across languages, since each of the 11 languages is thus evaluated on exactly the same content.

For both datasets, the results reported in this paper (similarly to those of [20]) are averages across the 10 random selections. Summary characteristics of our two datasets are reported in Table 1; excerpts from sample documents from the two datasets are displayed in Table 2.

4.2 Evaluation measures

To assess the model performance we employ F_1, the standard measure of text classification, and the more recently theorized K [55]. These two functions are defined as:

$$F_1 = \begin{cases} \dfrac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} & \text{if TP} + \text{FP} + \text{FN} > 0 \\ 1 & \text{if TP} = \text{FP} = \text{FN} = 0 \end{cases} \tag{7}$$

$$K = \begin{cases} \dfrac{\text{TP}}{\text{TP}+\text{FN}} + \dfrac{\text{TN}}{\text{TN}+\text{FP}} - 1 & \text{if TP} + \text{FN} > 0 \text{ and TN} + \text{FP} > 0 \\ 2\dfrac{\text{TN}}{\text{TN}+\text{FP}} - 1 & \text{if TP} + \text{FN} = 0 \\ 2\dfrac{\text{TP}}{\text{TP}+\text{FN}} - 1 & \text{if TN} + \text{FP} = 0 \end{cases} \tag{8}$$

where TP, FP, FN, TN represent the number of true positives, false positives, false negatives, and true negatives generated by a binary classifier. F_1 ranges between 0 (worst) and 1 (best) and is the harmonic mean of precision and recall, while K ranges between -1 (worst) and 1 (best).

To turn F_1 and K (whose definitions above are suitable for binary classification) into measures for multilabel classification, we compute their microaverages (F_1^μ and K^μ) and their macroaverages (F_1^M and K^M). F_1^μ and K^μ are obtained by first computing the class-specific values TP_j, FP_j, FN_j, TN_j, computing $\text{TP} = \sum_{j=1}^{|\mathcal{Y}|} \text{TP}_j$ (and analogously for FP, FN, TN), and then applying Equations 7 and 8. Instead, F_1^M and K^M are obtained by first computing the class-specific values of F_1 and K and then averaging them across all $y_j \in \mathcal{Y}$.

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.

4.3 Learners

Wherever possible, we use the same learner as used in [20], i.e., Support Vector Machines (SVMs) as implemented in the `scikit-learn` package.[9] 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, by testing all values $C = 10^i$ for $i \in \{-1, ..., 4\}$ by means of 5-fold cross-validation. We use Platt calibration in order to calibrate the 1st-tier classifiers used in the Posteriors VGF and (when using averaging as the aggregation policy) the classifiers that map document views into vectors of posterior probabilities. We employ the linear kernel for the 1st-tier classifiers used in the Posteriors VGF, and the RBF kernel (i) for the classifiers used for implementing the averaging aggregation policy, and (ii) for the 2nd-tier classifier.

In order to generate the BERT VGF (see Section 3.4), we rely on the pre-trained model released by `Huggingface`[10] [66]. For each run, we train the model following the settings suggested by Devlin et al. [17], i.e., we add one classification layer on top of the output of mBERT (the special token `[CLS]`) and fine-tune the entire model end-to-end by minimising the binary cross-entropy loss function. We use the AdamW optimizer [36] with the learning rate set to 1e-5 and the weight decay set to 0.01. We also set the learning rate to decrease by means of a scheduler (StepLR) with step size equal to 25 and gamma equal to 0.1. We set the training batch size to 4 and the maximum length of the input (in terms of tokens) to 512 (which is the maximum input length of the model). Given that the number of training examples in our datasets is comparatively smaller than that used in Devlin et al. [17], we reduce the maximum number of epochs to 50, and apply an early-stopping criterion that terminates the training after 5 epochs showing no improvement (in terms of F_1^M) in the validation set (a held-out split containing 20% of the training documents) in order to avoid overfitting. After convergence, we perform one last training epoch on the validation set.

Each of the experiments we describe is performed 10 times, on 10 different samples extracted from the dataset, in order to assess its statistical significance by means of the paired t-test mentioned in Section 3.6. All the results displayed in the tables included in this paper are averages across these 10 samples and across the $|\mathcal{L}|$ languages in the datasets.

[9] https://scikit-learn.org/stable/index.html
[10] We use the `bert-base-multilingual-cased` model available at https://huggingface.co/

We run all the experiments on a machine equipped with a 12-core processor Intel Core i7-4930K at 3.40GHz with 32GB of RAM under Ubuntu 18.04 (LTS) and Nvidia GeForce GTX 1080 equipped with 8GB of RAM.

4.4 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 [20], namely, *Lightweight Random Indexing* (LRI [43]), *Cross-Lingual Explicit Semantic Analysis* (CLESA [59]), *Kernel Canonical Correlation Analysis* (KCCA [63]), and *Distributional Correspondence Indexing* (DCI [42]). For all systems but gFun, the results we report are excerpted from [20], so we refer to that paper for the detailed setups of these baselines; the comparison is fair anyway, since our experimental setup is identical to that of [20].

We also include mBERT [17] as an additional baseline. In order to generate the mBERT baseline, we follow exactly the same procedure as described above for the BERT VGF. Note that the difference between mBERT and BERT VGF comes down to the fact that the former leverages a linear transformation of the document embeddings followed by a sigmoid activation in order to compute the prediction scores. On the other hand, BERT as a VGF is used as a feature extractor (or embedder). Once the document representations are computed (by mBERT), we project them to the space of the posterior probabilities via a set of SVMs. We also experiment with an alternative training strategy in which we simply train the classification layer, and leave the pre-trained parameters of mBERT untouched, but omit the results obtained using this strategy because in preliminary experiments it proved inferior to the other strategy by a large margin.

Similarly to [20] we also report an “idealized” baseline (i.e., one whose performance all CLC methods should strive to reach up to), called UpperBound, which consists of replacing each non-English training example by its corresponding English version, training a monolingual English classifier, and classifying all the English test documents. UpperBound is present only in the JRC-Acquis experiments since in RCV1/RCV2 the English versions of non-English training examples are not available.

4.5 Results of many-shot CLTC experiments

In this section we report the results that we have obtained in our many-shot CLTC experiments on the RCV1/RCV2 and JRC-Acquis datasets.[11] These experiments are run in “everybody-helps-everybody” mode, i.e., all training data, from all languages, contribute to the classification of all unlabelled data, from all languages.

We will use the notation -X to denote a gFun instantiation that uses only one VGF, namely the Posteriors VGF; gFun-X is thus equivalent to the original Fun architecture, but with the addition of the normalisation steps discussed in Section 3.6. Analogously, -M will denote the use of the MUSEs VGF (Section 3.2), -W the use of the WCEs VGF (Section 3.3), and -B the use of the BERT VGF (Section 3.4).

Tables 3 and 4 report the results obtained on RCV1/RCV2 and JRC-Acquis, respectively. We denote different setups of gFun by indicating after the hyphen the VGFs that the variant uses. For each dataset we report the results for 7 different baselines and 9 different configurations of gFun, as well as for two distinct evaluation metrics (F_1 and K) aggregated across the $|\mathcal{Y}|$ different classes by both micro- and macro-averaging.

The results are grouped in four batches of methods. The first one contains all baseline methods. The remaining batches present results obtained using a selection of meaningful combinations of VGFs: the 2nd batch reports the results obtained by gFun when equipped with one single VGF, the 3rd batch reports ablation results, i.e., results obtained by removing one VGF from a setting containing all VGFs, while in the last batch we report the results obtained by jointly using all the VGFs discussed.

[11]In an earlier, shorter version of this paper [45] we report different results for the very same datasets. The reason of the difference is that in [45] we use concatenation as the aggregation policy while we here use averaging.

The results clearly indicate that the fine-tuned version of multilingual BERT consistently outperforms all the other baselines, on both datasets. Concerning gFun's results, among the different settings of the second batch (testing different VGFs in isolation), the only configuration that consistently outperforms mBERT in RCV1/RCV2 is gFun-B. Conversely, on JRC-Acquis, all four VGFs in isolation manage to beat mBERT for at least 2 evaluation measures. Most other configurations of gFun we have tested (i.e., configurations involving more than one VGF) consistently beat mBERT, with the sole exception of gFun-XMW on RCV1/RCV2.

Method	F_1^M	F_1^μ	K^M	K^μ
Naïve	.467 ± .083	.776 ± .052	.417 ± .090	.690 ± .074
LRI [43]	.490 ± .077	.771 ± .050	.440 ± .086	.696 ± .069
CLESA [59]	.471 ± .074	.714 ± .061	.434 ± .080	.659 ± .075
KCCA [63]	.385 ± .079	.616 ± .065	.358 ± .088	.550 ± .073
DCI [42]	.485 ± .070	.770 ± .052	.456 ± .082	.696 ± .065
FUN [20]	.534 ± .066	.802 ± .041	.506 ±. 073	.760 ± .052
mBERT [16]	.581 ± .014	.817 ± .005	.559 ± .015	.788 ± .008
gFun–X	.547 ± .065	.798 ± .041	.551 ± .070	.799 ± .046
gFun–M	.548 ± .066	.769 ± .042	.564 ± .077	.765 ± .048
gFun–W	.487 ± .062	.743 ± .054	.511 ± .086	.730 ± .058
gFun–B	.608 ± .064‡	.826 ± .040†	**.603** ± .078	.797 ± .049
gFun–XMB	**.611** ± .068	**.833** ± .035	.597 ± .077‡	**.813** ± .045
gFun–XWB	.581 ± .062	.821 ± .037	.574 ± .073	.797 ± .046
gFun–XMW	.558 ± .061	.801 ± .038	.558 ± .072	.788 ± .046
gFun–WMB	.593 ± .065†	.821 ± .036	.582 ± .079†	.795 ± .048
gFun–XWMB	.596 ± .064†	.826 ± .035†	.579 ± .075†	.802 ± .046

Table 3. Many-shot CLTC results on the RCV1/RCV2 dataset. Each cell reports the mean value and the standard deviation across the 10 runs. **Boldface** indicates the best method overall, while greyed-out cells indicate the best method within the same group of methods. 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 a $0.05 \leq p$-value.

Something that jumps to the eye is that gFun-X yields better results than Fun, which is different from it only for the the normalisation steps of Section 3.6. This is a clear indication that these normalisation steps are indeed beneficial.

Combinations relying on WCEs seem to perform comparably better in the JRC-Acquis dataset, and worse in RCV1/RCV2. This can be ascribed to the fact that the amount of information brought about by word-class correlations is higher in the case of JRC-Acquis (since this dataset contains no fewer than 300 classes) than in RCV1/RCV2 (which only contains 73 classes). Notwithstanding this, the WCEs VGF seems to be the weakest among the VGFs that we have tested. Conversely, the strongest VGF seems to be the one based on mBERT, though it is also clear from the results that other VGFs contribute to further improve the performance of gFun; in particular, the combination gFun-XMB stands as the top performer overall, since it is always either the best performing model or a model no different from the best performer in a statistically significant sense.

Upon closer examination of Tables 3 and 4, the 2nd, 3rd, and 4th batches help us in highlighting the contribution of each signal (i.e., information brought about by the VGFs).

Let us start from the 4th batch, where we report the results obtained by the configuration of gFun that exploits all of the available signals (gFun-XWMB). In RCV1/RCV2 such a configuration yields superior results to the

Method	F_1^M	F_1^μ	K^M	K^μ
Naïve	.340 ± .017	.559 ± .012	.288 ± .016	.429 ± .015
LRI [43]	.411 ± .027	.594 ± .016	.348 ± .025	.476 ± .020
CLESA [59]	.379 ± .034	.557 ± .024	.330 ± .034	.453 ± .029
KCCA [63]	.206 ± .018	.357 ± .023	.176 ± .017	.244 ± .022
DCI [42]	.317 ± .012	.510 ± .014	.274 ± .013	.382 ± .016
Fun [20]	.399 ± .013	.587 ± .009	.365 ± .014	.490 ± .013
mBERT [16]	.420 ± .023	.608 ± .016	.379 ± .006	.507 ± .009
gFun–X	.432 ± .015	.587 ± .010	.441 ± .016	.553 ± .013
gFun–M	.440 ± .039	.586 ± .032	.442 ± .045	.549 ± .034
gFun–W	.410 ± .016	.553 ± .014	.410 ± .021	.525 ± .022
gFun–B	.501 ± .023	.627 ± .016	.485 ± .023	.574 ± .019
gFun–XMB	**.525** ± .020	**.649** ± .014	**.528** ± .023	**.620** ± .017
gFun–XWB	.497 ± .011	.621 ± .008	.508 ± .011	.606 ± .010
gFun–XMW	.475 ± .012	.604 ± .010	.489 ± .014	.593 ± .011
gFun–WMB	.513 ± .016	.632 ± .011	.522 ± .017‡	.619 ± .013‡
gFun–XWMB	.514 ± .014	.635 ± .010	.521 ± .015†	.618 ± .011‡
UpperBound	.599	.707	.547	.632

Table 4. As Table 3, but using JRC-Acquis instead of RCV1/RCV2.

single-VGF settings (note that even though results for gFun-B (.608) are higher than those for gFun-XWMB (.596), this difference is not statistically significant, with a p-value of .680, according to the two-tailed t-test that we have run). Such a result indicates that there is indeed a synergy among the heterogeneous representations.

In the 3rd batch, we investigate whether all of the signals are mutually beneficial or if there is some redundancy among them. We remove from the "full stack" (gFun-XWMB) one VGF at a time. The removal of the BERT VGF has the worst impact on F_1^M. This was expected since, in the single-VGF experiments, gFun-B was the top-performing setup. Analogously, by removing representations generated by the Posteriors VGF or those generated by the MUSEs VGF, we have a smaller decrease in F_1^M results. On the contrary, ditching WCEs results in a higher F_1^M score (our top-scoring configuration); the difference between gFun-XWMB and gFun-XMB is not statically significant in RCV1/RCV2 (with a p-value between 0.001 and 0.05), but it is significant in JRC-Acquis. This is an interesting fact: despite the fact that in the single-VGF setting the WCEs VGF is the worst-performing, we were not expecting its removal to be beneficial. Such a behaviour suggests that the WCEs are not well-aligned with the other representations, resulting in worse performance across all the four metrics. This is also evident if we look at results reported in [47]. If we remove from gFun-XMW (.558) the Posteriors VGF, thus obtaining gFun-MW, we obtain a F_1^M score of .536; by removing the MUSEs VGF, thus obtaining gFun-XW, we lower the F_1^M to .523; instead, by discarding the WCEs VGF, thus obtaining gFun-XM, we increase F_1^M to .575. This behaviour tells us that the information encoded in the Posteriors and WCEs representations is diverging: in other words, it does not help in building more easily separable document embeddings. Results on JRC-Acquis are along the same line.

In Figure 4, we show a more in-depth analysis of the results, in which we compare, for each language, the relative improvements obtained in terms of F_1^M (the other evaluation measures show similar patterns) by mBERT (the top-performing baseline) and a selection of gFun configurations, with respect to the Naïve solution.

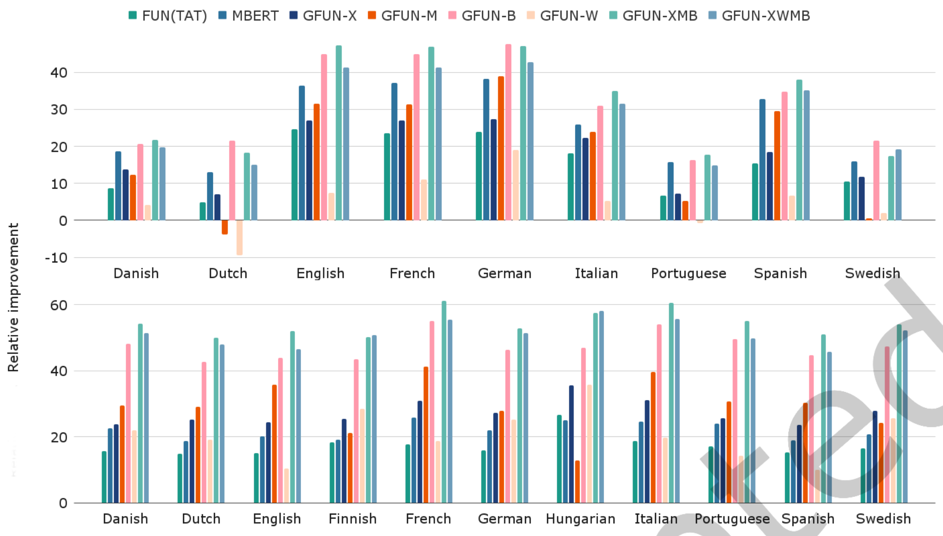

Fig. 4. Percentage of relative improvement per language obtained by different cross-lingual models in the many-shot CLTC experiments, in terms of F_1^M with respect to the Naïve solution, for RCV1/RCV2 (top) and JRC-Acquis (bottom).

These results confirm that the improvements brought about by gFun-X with respect to Fun are consistent across all languages, and not only as an average across them, for both datasets. The only configurations that underperform some monolingual naïve solutions (i.e., that have a *negative* relative improvement) are gFun-M (for Dutch) and gFun-W (for Dutch and Portuguese) on RCV1/RCV2. These are also the only configurations that sometimes fare worse than the original Fun. The configurations gFun-B, gFun-XMB, and gFun-XWMB, all perform better than the baseline mBERT on almost all languages and on both datasets (the only exception for this happens for Portuguese when using gFun-XWMB in RCV1/RCV2), with the improvements with respect to mBERT being markedly higher on JRC-Acquis. Again, we note that, despite the clear evidence that the VGF based on mBERT brings to bear the highest improvements overall, all other VGFs do contribute to improving the classification performance; the histograms of Figure 4 now reveal that the contributions are consistent across all languages. For example, gFun-XMB outperforms gFun-B for six out of nine languages in RCV1/RCV2, and for all eleven languages in JRC-Acquis.

As a final remark, we should note that the document representations generated by the different VGFs are certainly not entirely independent (although their degree of mutual dependence would be hard to measure precisely), since they are all based on the distributional hypothesis, i.e., on the notion that systematic co-occurrence (of words and other words, of words and classes, of classes and other classes, etc.) is an evidence of correlation. However, in data science, mutual independence is not a necessary condition for usefulness; we all know this, e.g., from the fact that the “bag of words” model of representing text works well despite the fact that it makes use of thousands of features that are not independent of each other. Our results show that, in the best-performing setups of gFun, several such VGFs coexist despite the fact that they are probably not mutually independent, which seems to indicate that the lack of independence of these VGFs is not an obstacle.

4.6 Results of zero-shot CLTC experiments

Fun was not originally designed for dealing with zero-shot scenarios since, in the absence of training documents for a given language, the corresponding first-tier language-dependent classifier cannot be trained. Nevertheless, Esuli et al. [20] managed to perform zero-shot cross-lingual experiments by plugging in an auxiliary classifier trained on MUSEs representations that is invoked for any target language for which training data are not available, provided that this language is among the 30 languages covered by MUSEs.

Instead, gFun caters for zero-shot cross-lingual classification *natively*, provided that at least one among the VGFs it uses is able to generate representations for the target language with no training data (for the VGFs described in this paper, this is the case of the MUSEs VGF and mBERT VGF for all the languages they cover). To see why, assume the gFun-XWMB instance of gFun using the averaging procedure for aggregation (Section 3.5). Assume that there are training documents for English, and that there are no training data for Danish. We train the system in the usual way (Section 2). For a Danish test document, the MUSEs VGF[12] and the mBERT VGF contribute to its representation, since Danish is one of the languages covered by MUSEs and mBERT. The aggregation function averages across all four VGFs (-XWMB) for English test documents, while it only averages across two VGFs (-MB) for Danish test documents. Note that the meta-classifier does not perceive differences between English test documents and Danish test documents since, in both cases, the representations it receives from the first tier come down to averages of calibrated (and normalized) posterior probabilities. Therefore, any language for which there are no training examples can be dealt with by our instantiation of gFun provided that this language is catered for by MUSEs and/or mBERT.

To obtain results directly comparable with the zero-shot setup employed by Esuli et al. [20], we reproduce their experimental setup. Thus, we run experiments in which we start with one single source language (i.e., a language endowed with its own training data), and we add new source languages iteratively, one at a time (in alphabetical order), until all languages for the given dataset are covered. At each iteration, we train gFun on the available source languages, and test on *all* the target languages. At the i-th iteration we thus have i source languages and $|\mathcal{L}|$ target (test) languages, among which i languages have their own training examples and the other $(|\mathcal{L}| - i)$ languages do not. For this experiment we choose the configuration involving all the VGFs (gFun-XWMB).

The results are reported in Figure 5 and Figure 6, where we compare the results obtained by Fun and gFun-XWMB on both datasets, for all our evaluation measures. Results are presented in a grid of three columns, in which the first one corresponds to the results of Fun as reported in [20], the second one corresponds to the results obtained by gFun-XWMB, and the third one corresponds to the difference between the two, in terms of absolute improvement of gFun-XWMB w.r.t. Fun. The results are arranged in four rows, one for each evaluation measure. Performance scores are displayed through heat-maps, in which columns represent target languages, and rows represent training iterations (with incrementally added source languages). Colour coding helps interpret and compare the results: we use red for indicating low values of accuracy and green for indicating high values of accuracy (according to the evaluation measure used) for the first and second columns; the third column (absolute improvement) uses a different colour map, ranging from dark blue (low improvement) to light green (high improvement). The tone intensities of the Fun and gFun colour maps for the different evaluation measures are independent of each other, so that the darkest red (resp., the lightest green) always indicates the worst (resp., the best) result obtained by any of the two systems *for the specific evaluation measure.*

Note that the lower triangular matrix within each heat map reports results for standard (many-shot) cross-lingual experiments, while all entries above the main diagonal report results for zero-shot cross-lingual experiments. As was to be expected, results for many-shots experiments tend to display higher figures (i.e., greener cells), while results for zero-shot experiments generally display lower figures (i.e., redder cells). These figures clearly

[12]In the absence of a proper training set, the IDF factor needed for computing the TFIDF weighting can be estimated using the test documents themselves, since TFIDF is an unsupervised weighting function.

		RCV1/RCV2				JRC-Acquis			
Method	Policy	F_1^M	F_1^μ	K^M	K^μ	F_1^M	F_1^μ	K^M	K^μ
gFun-XM	Concatenation	$0.562^\ddagger$	**0.806**	$0.552^\dagger$	$0.797^\ddagger$	0.468	0.610	0.466	0.572
gFun-XM	Averaging	**0.573**	$0.805^\ddagger$	**0.575**	**0.800**	**0.477**	**0.615**	$0.488^\ddagger$	0.588
gFun-XMW	Concatenation	0.540	0.791	0.530	0.773	0.461	0.609	0.445	0.560
gFun-XMW	Averaging	$0.558^\dagger$	$0.801^\dagger$	$0.558^\dagger$	0.788	$0.475^\ddagger$	0.604	**0.489**	**0.593**

Table 5. Results of many-shot CLTC experiments comparing the two aggregation policies on RCV1/RCV2 and JRC-Acquis (from [47]).

show the superiority of gFun over Fun, and especially so for the zero-shot setting, for which the magnitude of improvement is decidedly higher. The absolute improvement ranges from 18% of K^M to 28% of K^μ on RCV1/RCV2, and from 35% of F_1^M to 44% of K^μ in the case of JRC-Acquis.

In both datasets, the addition of new languages to the training set tends to help gFun improve the classification of test documents also for other languages for which a training set was already available anyway. This is witnessed by the fact that the green tonality of the columns in the lower triangular matrix becomes gradually darker; for example, in JRC-Acquis, the classification of test documents in Danish evolves stepwise from $K = 0.52$ (when the training set consists only of Danish documents) to $K = 0.62$ (when all languages are present in the training set).[13]

A direct comparison between the old and new variants of funnelling is conveniently summarized in Figure 7, where we display average values of accuracy (in terms of our four evaluation measures) obtained by each method across all experiments of the same type, i.e., standard cross-lingual (CLTC – values from the lower diagonal matrices of Figures 5 and 6) or zero-shot cross-lingual (ZSCLC – values from the upper diagonal matrices), as a function of the number of training languages, for both datasets. These histograms reveal that gFun improves over Fun in the zero-shot experiments. Interestingly enough, the addition of languages to the training set seems to have a positive impact in gFun, both for zero-shot and cross-lingual experiments.

4.7 Testing different aggregation policies

In this brief section we summarize the results of preliminary, extensive experiments in which we had compared the performance of different aggregation policies (concatenation vs. averaging); we here report only the results for the gFun-XM and gFun-XMW models (the complete set of experiments is described in [47]).

Table 5 reports the results we obtained for RCV1/RCV2 and JRC-Acquis, respectively. The results conclusively indicate that the averaging aggregation policy yields either the best results, or results that are not different (in a statistically significant sense) from the best ones, in all cases. This, along with other motivations discussed in Section 3.5 (scalability, and the fact that it enables zero-shot classification) makes us lean towards adopting averaging as the default aggregation policy.

Incidentally, Table 5 also seems to indicate that WCEs work better in JRC-Acquis than in RCV1/RCV2. This is likely due to the fact that, as observed in [44], the benefit brought about by WCEs tends to be more substantial when the number of classes is higher, since a higher number of classes means that WCEs have a higher dimensionality, and that they thus bring more information to the process.

[13]That the addition of new languages to the training set helps improve the classification of test documents for other languages for which a training set was already available, is true also in Fun. However, this does not emerge from Figure 5 and Figure 6 (which are taken from [20]). This has already been noticed by Esuli et al. [20], who argue that this happens only in the zero-shot version of Fun, and is due to the zero-shot classifier's failure to deliver well calibrated probabilities.

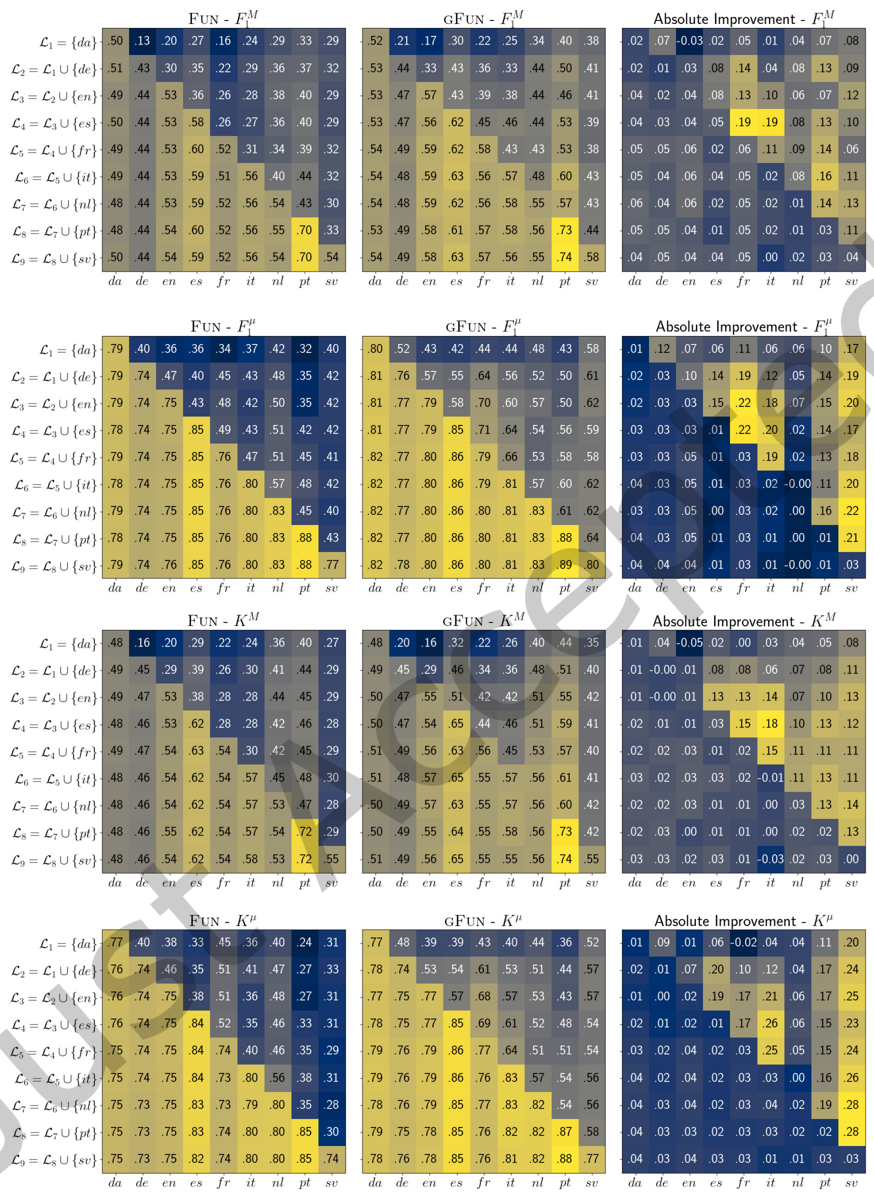

Fig. 5. Results of zero-shot CLTC experiments on RCV1/RCV2

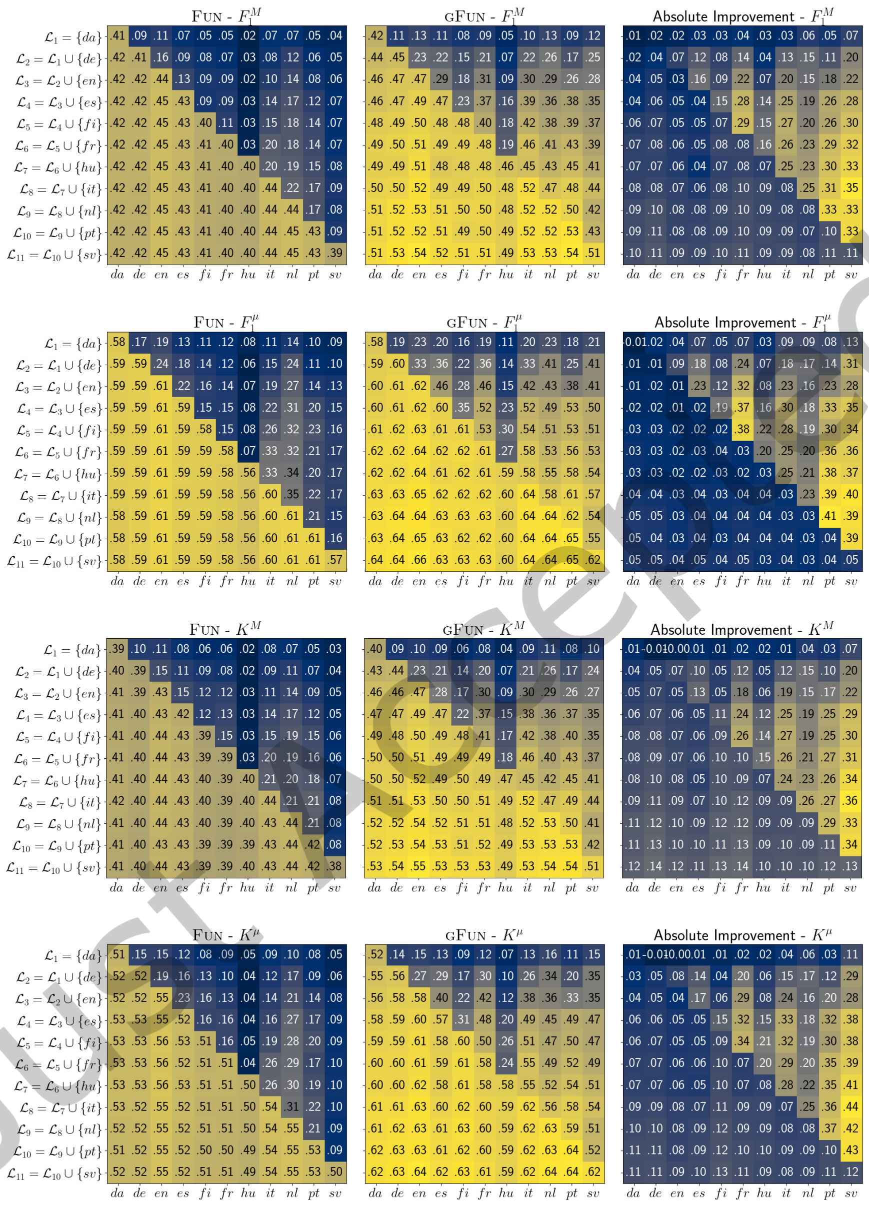

Fig. 6. Results of zero-shot CLTC experiments on JRC-Acquis

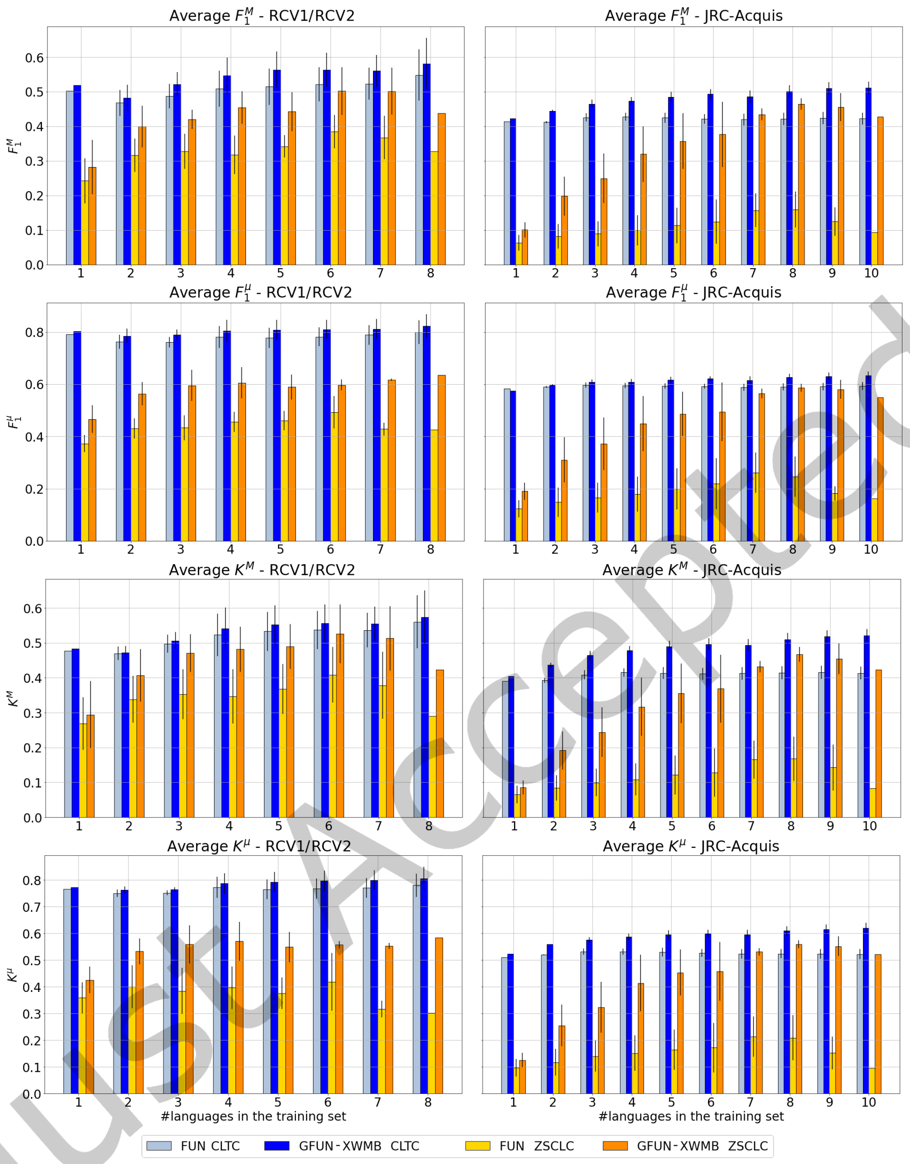

Fig. 7. Performance of different CLTC systems as a function of the number of language-specific training sets used.

4.8 Learning-Curve Experiments

In this section we report the results obtained in additional experiments aiming to quantify the impact on accuracy of variable amounts of target-language training documents. Given the supplementary nature of these experiments, we limit them to the RCV1/RCV2 dataset. Furthermore, for computational reasons we carry out these experiments only on a subset of the original languages (namely, English, German, French, and Italian). In Figure 8 we report the results, in terms of F_M^1, obtained on RCV1/RCV2. For each of the 4 languages we work on, we assess the performance of gFun-XMB by varying the amount of target-language training documents; we carry out experiments with 0%, 10%, 20%, 30%, 50%, and 100% of the training documents. For example, the experiments on French (Figure 8, bottom left) are run by testing on 100% of the French test data a classifier trained with 100% of the English, German, and Italian training data and with variable proportions of the French training data. We compare the results with those obtained (using the same experimental setup) by the Naïve approach (see Section 1 and 4.1) and by Fun[20].

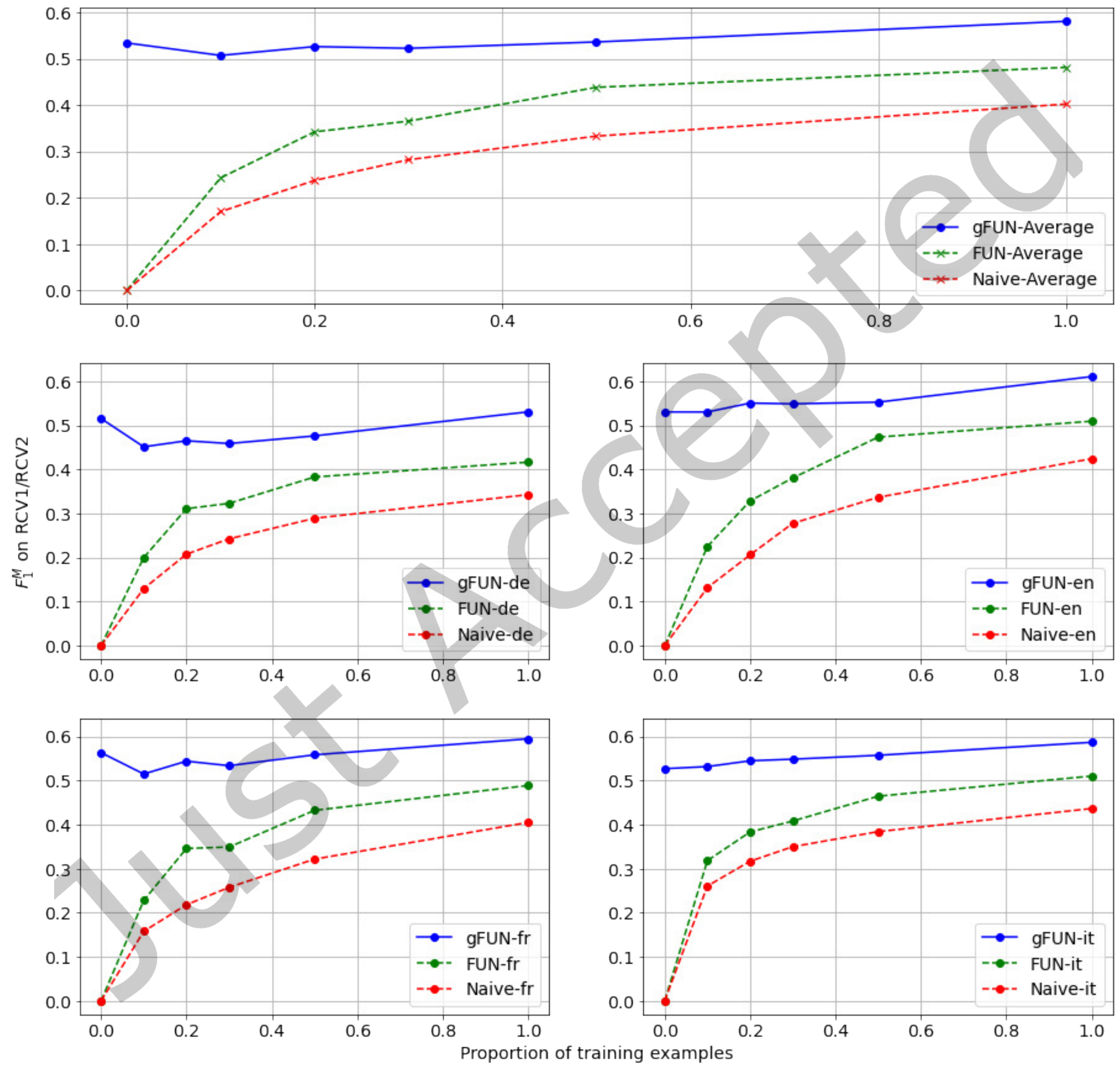

Fig. 8. Learning-curve experiments performed on RCV1/RCV2 dataset. Experiments are performed for increasing proportions of training examples (i.e., for 0%, 10%, 20%, 30%, 50%, 100%) for four languages (i.e., German, English, French, and Italian). The configuration of gFun deployed is gFun-XMB. We compare the performance of gFun-XMB with that displayed by FUN [20] and by the Naïve approach.

It is immediate to note from the plots that the two baseline systems have a very low performance when there are few target-language training examples, but this is not true for gFun-WMB, which has a very respectable performance even with 0% target-language training examples; indeed, gFun-WMB is able to almost bridge the gap between the zero-shot and many-shot settings, i.e., for gFun-WMB the difference between the F_M^1 values obtained with 0% or 100% target-language training examples is moderate. On the contrary, for the two baseline systems considered, the inclusion of additional target-language training examples results in a substantial increase in performance; however, both baselines substantially underperform gFun-WMB, for any percentage of target-language training examples, and for each of the 4 target languages.

4.9 Precision and recall

In this section we look at precision and recall for individual languages, as obtained by gFun, with the goal of investigating if any significant language-specific pattern emerges.

Figure 9 and 10 display precision and recall (in both their macro- and micro-averaged versions) obtained for the best-performing setting (-XWB) of gFun, in one run on RCV1/RCV2 (Figure 9) and one run on JRC-Acquis (Figure 10).

The main observation that can be made by observing these figures is that, for each language and for each dataset, average precision is always invariably higher than average recall. This can be explained by the fact that all our datasets are imbalanced at the class level (i.e., for each class the positives are far outnumbered by the negatives). In these cases, it is well-known that a learner that optimizes for vanilla accuracy (or for a proxy of it, such as the hinge loss, which is our case) tends to err on the side of caution (i.e., choose a high decision threshold); after all, on a test set in which, say, 99% of the examples are negatives, classifying all the unlabelled examples as negatives (which is the result of an extremely high decision threshold) rewards the classifier with an extremely high value of vanilla accuracy, i.e., 0.99. In other words, imbalanced data plus hinge loss as the loss to minimize means high decision threshold, which in turn means, quite obviously, higher precision and lower recall. As mentioned above, this tendency is displayed essentially by all languages and for both datasets.

5 LEARNING ALTERNATIVE COMPOSITION FUNCTIONS: THE RECURRENT VGF

The embeddings-based VGFs that we have described in Sections 3.2 and 3.3 implement a simple dot product as a means for deriving document embeddings from the word embeddings and the TFIDF-weighted document vector. However, while such an approach is known to produce document representations that perform reasonably well on short texts [14], there is also evidence that more powerful models are needed for learning more complex "composition functions" for texts [12, 58]. In NLP and related disciplines, *composition functions* are defined as functions that take as input the constituents of a sentence (sometimes already converted into distributed dense representations), and output a single vectorial representation capturing the overall semantics of the given sentence. In this section, we explore alternatives to the dot product for the VGFs based on MUSEs and WCE.

For this experiment, for generating document embeddings we rely on recurrent neural networks (RNNs). In particular, we adopt the *gated recurrent unit* (GRU) [10], a lightweight variant of the *long-short term memory* (LSTM) unit [26], as our recurrent cell. GRUs have fewer parameters than LSTMs and do not learn a separate output function (such as the output gate in LSTMs), and are thus more efficient during training. (In preliminary experiments we have carried out, we have found no significant differences in performance between GRU and LSTM; the former is much faster to train, though.) This gives rise to what we call the *Recurrent VGF*.

In the Recurrent VGF we thus infer the composition function at VGF fitting time. During the training phase, we train an RNN to generate good document representations from a set of language-aligned word representations consisting of the concatenation of WCEs and MUSEs. This VGF is trained in an end-to-end fashion. The output

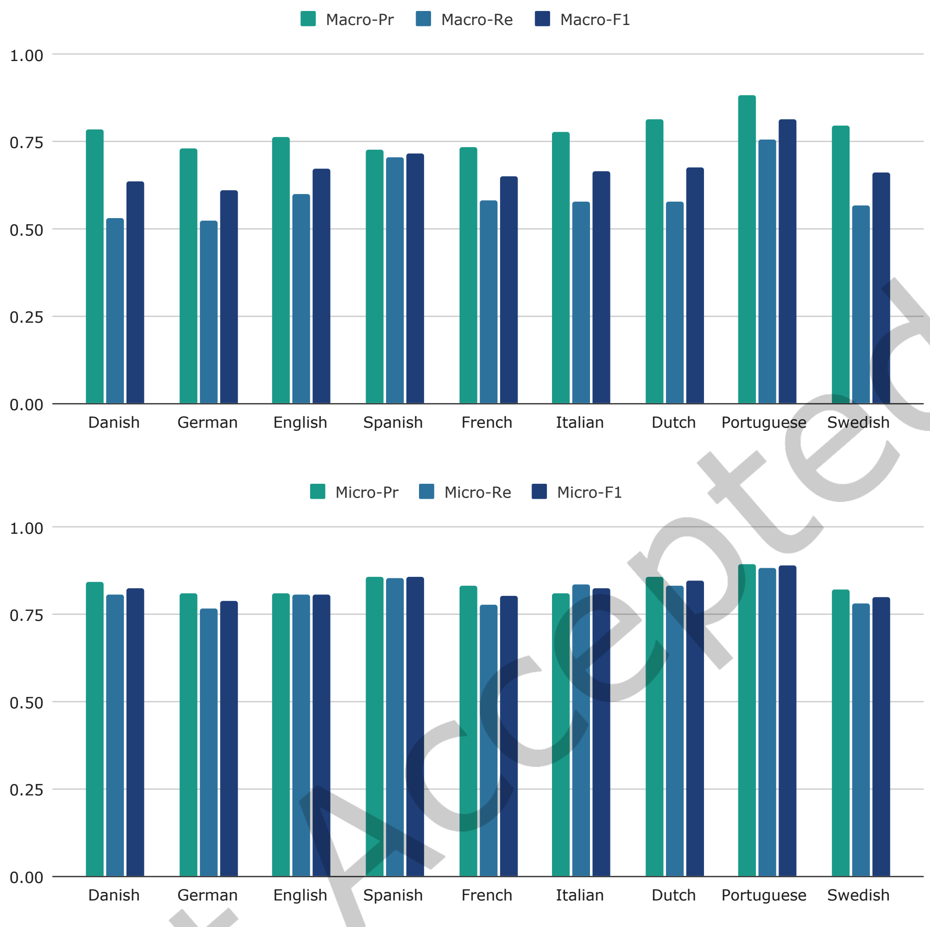

Fig. 9. Precision and Recall (both in their micro- and macro-averaged version) for each of the 9 different languages in RCV1/RCV2. Results are computed on a single run.

representations of the training documents generated by the GRU are projected onto a $|\mathcal{Y}|$-dimensional space of label predictions; the network is trained by minimising the binary cross-entropy loss between the predictions and the true labels. We explore different variants depending on how the parameters of the embedding layer are initialized (see below). We do not freeze the parameters of the embedding layers, so as to allow the optimisation procedure to fine-tune the embeddings. We use the Adam optimizer [32] with initial learning rate set at 1e-3 and no weight decay. We halve the learning rate every 25 epochs by means of StepLR (gamma = 0.5, step size = 25). We set the training batch size to 256 and compute the maximum length of the documents dynamically at each batch by taking their average length. Documents exceeding the computed length are truncated, whereas shorter ones are padded. Finally, we train the model for a maximum of 250 epochs, with an early-stopping criterion that terminates the training after 25 epochs with no improvement on the validation F_1^M.

There is only one Recurrent VGF in the entire gFun architecture, which processes all documents, independently of the language they belong to. Once trained, the last linear layer is discarded. All training documents are then

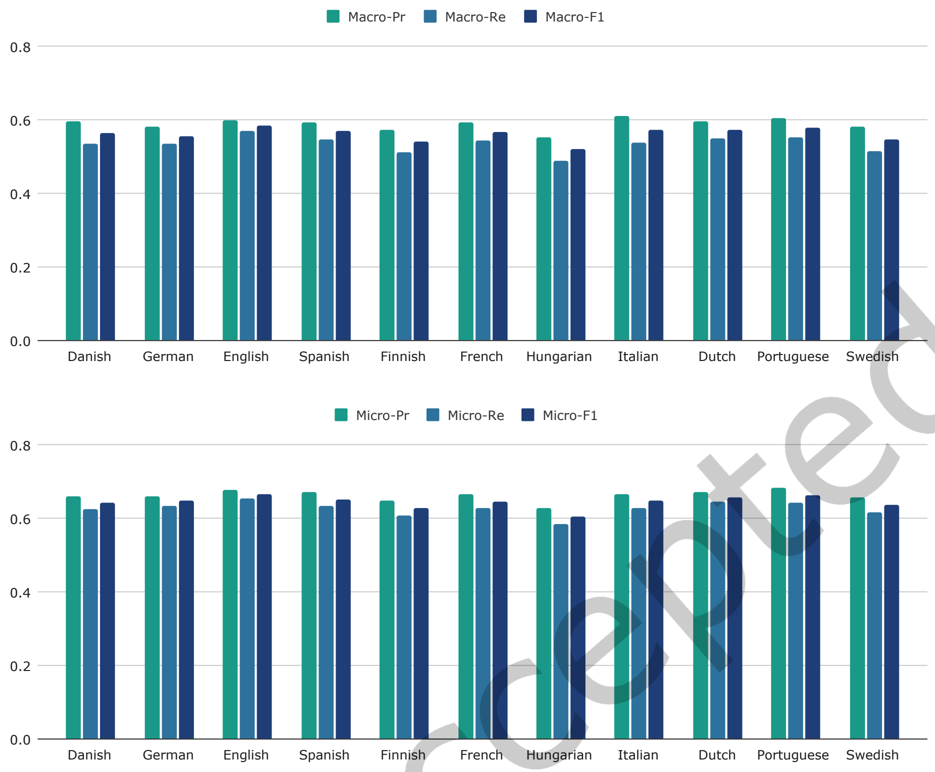

Fig. 10. Precision and Recall (both in their micro- and macro-averaged version) for each of the 11 different languages in JRC-Acquis. Results are computed on a single run.

passed through the GRU and converted into document embeddings, which are eventually used to train a calibrated classifier which returns posterior probabilities for each class in the codeframe.

5.1 Experiments

We perform many-shot CLTC experiments using the Recurrent VGF trained on MUSEs only (denoted -R_M), or trained on the concatenation of MUSEs and WCEs (denoted -R_{MW}). We do not explore the case in which the GRU is trained exclusively on WCEs since, as explained in [44], WCEs are meant to be concatenated to general-purpose word embeddings. Similarly, we avoid exploring combinations of VGFs based on redundant sources of information, e.g., we do not attempt to combine the MUSEs VGFs with the Recurrent VGF, since this latter already makes use of MUSEs.

Tables 6 and 7 report on the experiments we have carried out using the Recurrent VGF, in terms of all our evaluation measures, for RCV1/RCV2 and JRC-Acquis, respectively. These results indicate that the Recurrent VGF under-performs the dot product criterion (this can be easily seen by comparing each result with its counterpart in Tables 3 and 4). A possible reason for this might be the fact that the amount of training documents available in our experimental setting is insufficient for learning a meaningful composition function. A further possible reason might be the fact that, in classification by topic, the mere presence or absence of certain predictive words

Method	F_1^M	F_1^μ	K^M	K^μ
gFun–R$_M$	.439 ± .072	.717 ± .067	.450 ± .091	.692 ± .071
gFun–R$_{MW}$	.431 ± .086	.731 ± .064	.411 ± .102	.665 ± .081
gFun–BR$_M$	.566 ± .065	.810 ± .040	.559 ± .083	.774 ± .050
gFun–BR$_{MW}$	.581 ± .064	.813 ± .039	.582 ± .080†	.794 ± .049
gFun–XR$_{MW}$	.527 ± .060	.788 ± .042	.531 ± .073	.777 ± .049
gFun–XBR$_M$	**.603** ± .066	**.826** ± .038	**.601** ± .077	**.811** ± .046
gFun–XBR$_{MW}$	.581 ± .059	.815 ± .037	.583 ± .074†	.799 ± .047

Table 6. Cross-lingual text classification results on RCV1/RCV2 dataset. Tests of statistical significance are performed against the best results found in Table 3.

Method	F_1^M	F_1^μ	K^M	K^μ
gFun–R$_M$	.225 ± .074	.379 ± .096	.234 ± .076	.354 ± .096
gFun–R$_{MW}$	.314 ± .019	.488 ± .022	.281 ± .020	.393 ± .024
gFun–BR$_M$	.390 ± .027	.561 ± .021	.358 ± .027	.466 ± .021
gFun–BR$_{MW}$	.470 ± .017	.598 ± .013	.472 ± .020	.564 ± .018
gFun–XR$_{MW}$	.418 ± .011	.569 ± .008	.423 ± .012	.528 ± .010
gFun–XBR$_M$	**.501** ± .016	**.634** ± .011	**.501** ± .020	**.595** ± .016
gFun–XBR$_{MW}$	.483 ± .011	.615 ± .008	.482 ± .014	.577 ± .011

Table 7. As Table 6, but using JRC-Acquis instead of RCV1/RCV2.

captures most of the information useful for determining the correct class labels, while the information conveyed by word order is less useful, or too difficult to capture. In future work it might thus be interesting to test the Recurrent VGF on tasks other than classification by topic.

Another aspect that jumps to the eye is that the relative improvements brought about by the addition of WCEs tend to be larger in JRC-Acquis than in RCV1/RCV2 (in which the presence of WCEs is sometimes detrimental). This is likely due to the fact that JRC-Acquis has more classes, something that ends up enriching the representations of WCEs. Somehow surprisingly, though, the best configuration is one not equipped with WCEs (and this happens also for JRC-Acquis). This might be due to a redundancy of the information captured by WCEs with respect to the information already captured in the other views. In the future, it might be interesting to devise ways for distilling the novel information that a VGF could contribute to the already existing views, and discarding the rest during the aggregation phase.

6 RELATED WORK

The first published paper on CLTC is [6]; in this work, as well as in [22], the task is tackled by means of a bag-of-words representation approach, whereby the texts are represented as standard vectors of length $|\mathcal{V}|$, with $\mathcal{V}$ being the union of the vocabularies of the different languages. Transfer is thus achieved only thanks to features shared across languages, such as proper names.

Years later, the field started to focus on methods originating from *distributional semantic models* (DSMs) [34, 52, 53]. These models are based on the so-called "distributional hypothesis", which states that similarity in meaning results in similarity of linguistic distribution [25]. Originally, these models [18, 41] made use of *latent semantic* *analysis* (LSA) [15], which factors a term co-occurrence matrix by means of low-rank approximation techniques such as SVD, resulting in a matrix of principal components, where each dimension is linearly independent of the others. The first examples of cross-lingual representations were proposed during the '90s. Many of these early works relied on abstract linguistic labels, such as those from *discourse representation theory* (DRT) [30], instead of on purely lexical features [2, 54]. Early approaches were based on the construction of high-dimensional context-counting vectors where each dimension represented the degree of co-occurrence of the word with a specific word in one of the languages of interest. However, these original implementations of DSMs required to explicitly compute the term co-occurrence matrix, making these approaches unfeasible for large amounts of data.

A seminal work is that of Mikolov et al. [39], who first noticed that continuous word embedding spaces exhibit similar topologies across different languages, and proposed to exploit this similarity by learning a linear mapping from a source to a target embedding space, exploiting a parallel vocabulary for providing anchor points for learning the mapping. This has spawned several studies on cross-lingual word embeddings [4, 21, 67]; however, all these methods relied on external manually generated resources (e.g., multilingual seed dictionaries, parallel corpora, etc.). This is a severe limitation, since the quality of the resulting word embeddings (and the very possibility to generate them) relies on the availability, and the quality, of these external resources [35].

Machine Translation (MT) represents a natural direct solution to CLTC tasks. Unfortunately, when it comes to low-resource languages, MT systems may be either not available or not sufficiently effective. Nevertheless, the MT-based approach will presumably become more and more viable as the field of MT progresses: recently, Isbister et al. [28] have shown evidence that relying on MT in order to translate documents from low-resource languages to higher-resource languages (e.g., English) for which state-of-the-art models are available, is indeed preferable to multilingual solutions.

Pre-trained word-embeddings [7, 40, 48] have been a major breakthrough for NLP and have become a key component of most natural language understanding architectures. As of today, many methods developed for CLTC rely on pre-trained *cross-lingual word embeddings* [5, 11, 39, 56] (for a more in-depth review on the subject see [51]). These embeddings strive to map representations from one language to the other via different techniques (e.g., Procrustes alignment), thus representing different languages in different, but aligned, vector spaces. For example, [8, 68] exploit aligned word embeddings in order to successfully transfer knowledge from one language to another. The approach proposed in [8] is a hybrid parameter-based / feature-based method to CLTC, in which a set of convolutional neural networks is trained on both source and target texts, encoded via aligned word representations (namely, MUSEs [11]) while sharing kernel parameters to better identify the common features across different languages. Furthermore, the authors insert in the loss function a regularisation term based on maximum mean discrepancy [23] in order to encourage representations that are domain-invariant.

Standard word embeddings have recently been called *static* (or *global*) representations. This is because they do not take into account the context of usage of a word, thus allowing only a single context-independent representation for each word; in other words, the different meanings of polysemous words are collapsed into a single representation. By contrast, *contextual* word embeddings [17, 37, 38, 49] associate each word occurrence with a representation that is a function of the entire sequence in which the word appears. Before processing each word with the "contextualising" function, tokens are mapped to a primary static word representation by means of a *language model*, typically implemented by a transformer architecture previously trained on large quantities of textual data. This has yielded a shift in the way we operate with embedded representations, from a setting in which pre-trained embeddings were used to initialize the embedding layer of a deep architecture that is later fully trained, to another in which the representation of words, phrases, and documents, is carried out by the transformer; what is left for training entails nothing more than learning a prediction layer, and possibly fine-tuning the transformer for the task at hand.

Such a paradigm shift has fuelled the appearance of models developed (or adapted) to deal with multilingual scenarios. Current multilingual models are large architectures directly trained on several languages at once, i.e., are models in which multilingualism is imposed by constraining all languages to share the same model parameters [17, 19, 33]. Given their extensive multilingual pre-training, such models are almost ubiquitous components of CLTC solutions.

For example, Zhang et al. [68] rely on pre-trained multilingual BERT in order to extract word representations aligned between the source and the target language. In a multitask-learning fashion, two identical-output (linear) classifier sare set up: the first is optimized on the source language via cross-entropy loss, while the second (i.e., the auxiliary classifier) is instead set to maximize the *margin disparity discrepancy* [70]. This is achieved by driving the auxiliary classifier to maximally differ (in terms of predictions) from the main classifier when applied to the target language, while returning similar predictions on the source language.

Guo et al. [24] tackle mono-lingual TC by exploiting multilingual data. They do so by using a contrastive learning loss as applied to Chinese BERT, a pre-trained (monolingual) language model. Then a unified model, which is composed of two trainable pooling layers and two auto-encoders, is trained on the union of the training data coming from both the source and the target languages. It is important to note that such a parameter-based approach requires parallel training data in order to successfully train the auto-encoders (i.e., so that they are able to create representations shared between the source and the target languages).

Karamanolakis et al. [31] propose a parameter-based approach. They first train a classifier on the source language, and then leverage the learned parameters of a set of b "seed" words to initialize the target language model (where b refers to the number of words that can be translated to the target language by a translation oracle). Subsequently, this model is used as a *teacher*, in knowledge-distillation fashion, to train a *student* classifier which is able to generalize beyond the b words transferred from the source classifier to the target classifier.

Wang et al. [65] leverage graph convolutional networks (GCNs) to integrate heterogeneous information within the task. They create a graph with the help of external resources such as a machine translation oracle and a POS-tagger. In the constructed graph, documents and words are treated as nodes, and edges are defined according to different relations, such as part-of-speech roles, semantic similarity, and document translations. Documents and words are connected by their co-occurrences, and the edges are labelled with their respective POSs. Document-document edges are also defined according to document-document similarity, as well as between translation equivalents. Once the heterogeneous cross-lingual graph is constructed, GCNs are applied in order to calculate higher-order representations of nodes with aggregated information. Finally, a linear transformation is applied to the document components in order to compute the prediction scores.

van der Heijden et al. [60] demonstrates the effectiveness of meta-learning approaches to cross-lingual text classification. Their goal is to create models that can adapt to new domains rapidly from few training examples. They propose a modification to MAML (Model-Agnostic Meta-Learning) called ProtoMAMLn. MAML is a meta-learning approach that optimises the base learner on the so-called "query set" (i.e., in-domain samples) after it has been updated on the so-called "support set" (that is, out-of-domain samples). ProtoMAMLn is an adaptation of ProtoMAML, where prototypes (computed by "Prototypical Network" [57]) are also L2-normalized.

Unlike our system, all the previously discussed approaches are designed to deal with a single source language only. Nevertheless, as we have already specified in Section 1, a solution designed to natively deal with multiple sources would be helpful. A similar idea is presented in [9], where the authors propose a method that relies on an initial multilingual representation of the document constituents. The model focuses on learning, on the one hand, a private (invariant) representation via an adversarial network, and on the other one, a common (language-specific) representation via a mixture-of-experts model. We do not include the system of [9] as a baseline in our experiments since it was designed to dealing with single-label problems.

7 CONCLUSIONS

We have presented Generalized Funnelling (gFun), a novel hierarchical learning ensemble method for heterogeneous transfer learning, and we have applied it to the task of cross-lingual text classification. gFun is an extension of Funnelling (Fun), an ensemble method where 1st-tier classifiers, each working on a different and language-dependent feature space, return a vector of calibrated posterior probabilities (with one dimension for each class) for each document, and where the final classification decision is taken by a meta-classifier that uses this vector as its input, and that can thus exploit class-class correlations. gFun extends Fun by allowing 1st-tier components to be arbitrary view-generating functions, i.e., language-dependent functions that each produce a language-agnostic representation ("view") of the document. In the instance of gFun that we have described here, for each document the meta-classifier receives as input a vector of calibrated posterior probabilities (as in Fun) aggregated to other embedded representations of the document that embody other types of correlations, such as word-class correlations (as encoded by "word-class embeddings"), word-word correlations (as encoded by "multilingual unsupervised or supervised embeddings"), and correlations between contextualized words (as encoded by multilingual BERT). In experiments carried out on two large, standard datasets for multilingual multilabel text classification, we have shown that this instance of gFun substantially improves over Fun, and over other strong baselines such as multilingual BERT itself. An additional advantage of gFun is that it is much better suited to zero-shot classification than Fun, since in the absence of training examples for a given language, views of the test document different from the one generated by a trained classifier can be brought to bear.

Aside from its very good classification performance, gFun has the advantage of having a "plug-and-play" character, since it allows arbitrary types of view-generating functions to be plugged into the architecture. A common characteristic in recent CLTC solutions is to leverage some kind of available, pre-trained cross- or multilingual resource; nevertheless, to the best of our knowledge, a solution trying to capitalise on multiple different (i.e., heterogeneous) resources has not yet been proposed. Furthermore, most approaches aim at improving the performance on the target language by exploiting a single source language (i.e., they are single-source approaches). In this, gFun differs from the discussed solutions since (i) it fully capitalises on multiple, heterogeneous available resources, (ii) while capable in principle to deal with single-source settings, it is especially designed to be deployed in multi-source settings and (iii) it is an "everybody-helps-everybody" solution, meaning that each language-specific training set contributes to the classification of all the documents, irrespectively of their language, and that all the languages benefit from the inclusion of other languages in the training phase (in other words, all the languages play both the role of source and target at the same time).

Finally, we note that gFun is a completely general-purpose heterogeneous transfer learning architecture, and its application (once appropriate VGFs are deployed) is not restricted to cross-lingual settings, or even to scenarios where text is involved. Indeed, in our future work we plan to test its adequacy to cross-media applications, i.e., situations in which the domains across which knowledge is transferred are represented by different media (say, text and images).

ACKNOWLEDGEMENTS

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 SoBigdata++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1, and by the AI4Media project, funded by the European Commission (Grant 951911) under the H2020 Programme ICT-48-2020. The authors' opinions do not necessarily reflect those of the European Commission.

REFERENCES

[1] Rie K. Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research* 6 (2005), 1817–1853.

[2] Chinatsu Aone and Douglas McKee. 1993. A language-independent anaphora resolution system for understanding multilingual texts. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL 1993)*. Columbus, US, 156–163. https://doi.org/10.3115/981574.981595

[3] 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.

[4] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 14th Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*. Austin, US, 2289–2294. https://doi.org/10.18653/v1/D16-1250

[5] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*. Vancouver, CA, 451–462. https://doi.org/10.18653/v1/P17-1042

[6] Nuria Bel, Cornelis H. Koster, and Marta Villegas. 2003. Cross-lingual text categorization. In *Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2003)*. Trondheim, NO, 126–139. https://doi.org/10.1007/978-3-540-45175-4_13

[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3 (2003), 1137–1155.

[8] Guan-Yuan Chen and Von-Wun Soo. 2019. Deep domain adaptation for low-resource cross-lingual text classification tasks. In *Proceedings of the 16th International Conference of the Pacific Association for Computational Linguistics (PACLING 2019)*. Hanoi, VN, 155–168.

[9] Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. 2019. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)*. Firenze, IT, 3098–3112. https://doi.org/10.18653/v1/P19-1299

[10] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 12th Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. Doha, QA, 1724–1734. https://doi.org/10.3115/v1/D14-1179

[11] 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.

[12] Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel Gershman, and Noah D. Goodman. 2018. Evaluating compositionality in sentence embeddings. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society (CogSci 2018)*. Madison, US.

[13] Oscar Day and Taghi M. Khoshgoftaar. 2017. A survey on heterogeneous transfer learning. *Journal of Big Data* 4 (2017), Article 17 (1–42). https://doi.org/10.1186/s40537-017-0089-0

[14] Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters* 80 (2016), 150–156.

[15] Scott C. Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (1990), 391–407.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Multilingual BERT readme document. https://github.com/google-research/bert/blob/a9ba4b8d7704c1ae18d1b28c56c0430d41407eb1/multilingual.md

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2019)*. Minneapolis, US, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[18] Susan T. Dumais, Todd A. Letsche, Michael L. Littman, and Thomas K. Landauer. 1997. Automatic cross-language retrieval using latent semantic indexing. In *Working Notes of the AAAI Spring Symposium on Cross-language Text and Speech Retrieval*. Stanford, US, 18–24. https://doi.org/10.1007/978-1-4615-5661-9_5

[19] Julian Eisenschlos, Sebastian Ruder, Piotr Czapla, Marcin Kardas, Sylvain Gugger, and Jeremy Howard. 2019. MultiFiT: Efficient Multi-lingual Language Model Fine-tuning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*. Hong Kong, CN, 5701–5706. https://doi.org/10.18653/v1/D19-1572

[20] 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

[21] Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*. Gothenburg, SE, 462–471.

https://doi.org/10.3115/v1/e14-1049
[22] Juan José García Adeva, Rafael A. Calvo, and Diego López de Ipińa. 2005. Multilingual approaches to text categorisation. *European Journal for the Informatics Professional* 5, 3 (2005), 43–51.
[23] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *Journal of Machine Learning Research* 13 (2012), 723—-773.
[24] Zhiqiang Guo, Zhaoci Liu, Zhenhua Ling, Shijin Wang, Lingjing Jin, and Yunxia Li. 2020. Text classification by contrastive learning and cross-lingual data augmentation for Alzheimer's disease detection. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING 2020)*. Barcelona, ES, 6161–6171.
[25] Zellig S. Harris. 1954. Distributional structure. *Word* 10, 23 (1954), 146–162. https://doi.org/10.1007/978-94-009-8467-7_1
[26] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
[27] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*. Lille, FR, 448–456.
[28] Tim Isbister, Fredrik Carlsson, and Magnus Sahlgren. 2021. Should we stop training more monolingual models, and simply use machine translation instead? In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa 2021)*. Reykjavik, IS, 385–390.
[29] Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (COLING 2020)*. Barcelona, ES, 6282–6293. https://doi.org/10.18653/v1/2020.acl-main.560
[30] Hans Kamp. 1988. Discourse representation theory: What it is and where it ought to go. *Natural Language at the Computer* 320, 1 (1988), 84–111.
[31] Giannis Karamanolakis, Daniel Hsu, and Luis Gravano. 2020. Cross-lingual text classification with minimal resources by transferring a sparse teacher. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*. Online Event, 3604–3622. https://doi.org/10.18653/v1/2020.findings-emnlp.323
[32] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*. San Diego, US.
[33] Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*. Vancouver, CA, 7057–7067.
[34] Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review* 104, 2 (1997), 211–240.
[35] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3 (2015), 211–225.
[36] Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*. New Orleans, US.
[37] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*. Long Beach, US, 6294–6305.
[38] Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. Context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th Conference on Computational Natural Language Learning (CoNLL 2016)*. Berlin, DE, 51–61. https://doi.org/10.18653/v1/K16-1006
[39] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. (2013). arXiv:1309.4168.
[40] Tomas Mikolov, Wen-Tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2013)*. Atlanta, US, 746–751.
[41] David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of the 7th Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*. Singapore, SN, 880–889. https://doi.org/10.3115/1699571.1699627
[42] 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
[43] 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
[44] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. 2021. Word-class embeddings for multiclass text classification. *Data Mining and Knowledge Discovery* 353, 3 (2021), 911–963. https://doi.org/10.1007/s10618-020-00735-3
[45] Alejandro Moreo, Andrea Pedrotti, and Fabrizio Sebastiani. 2021. Heterogeneous document embeddings for cross-lingual text classification. In *Proceedings of the 36th ACM Symposium on Applied Computing (SAC 2021)*. Gwangju, KR, 685–688. https://doi.org/10.1145/3412841.3442093

[46] Nikolaos Pappas and James Henderson. 2019. GILE: A generalized input-label embedding for text classification. *Transactions of the Association for Computational Linguistics* 7 (2019), 139–155.

[47] Andrea Pedrotti. 2020. *Heterogeneous document embeddings for multi-lingual text classification.* Master's thesis. University of Pisa, Pisa, IT.

[48] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 12th Conference on Empirical Methods in Natural Language Processing (EMNLP 2014).* Doha, QA, 1532–1543.

[49] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).* New Orleans, US, 2227–2237. https://doi.org/10.18653/v1/N18-1202

[50] John C. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*, Alexander Smola, Peter Bartlett, Bernard Schölkopf, and Dale Schuurmans (Eds.). The MIT Press, Cambridge, MA, 61–74.

[51] Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research* 65, 1 (2019), 569–630. https://doi.org/10.1613/jair.1.11640

[52] Magnus Sahlgren. 2006. *The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.* Ph. D. Dissertation. Swedish Institute for Computer Science, University of Stockholm, Stockholm, SE.

[53] Hinrich Schütze. 1993. Word space. In *Proceedings of the 6th Conference on Neural Information Processing Systems (NIPS 1993).* Denver, US, 895–902.

[54] Tanja Schultz and Alex Waibel. 2001. Language-independent and language-adaptive acoustic modeling for speech recognition. *Speech Communication* 35, 1 (2001), 31–51. https://doi.org/10.1016/S0167-6393(00)00094-7

[55] Fabrizio Sebastiani. 2015. An axiomatically derived measure for the evaluation of classification algorithms. In *Proceedings of the 5th ACM International Conference on the Theory of Information Retrieval (ICTIR 2015).* Northampton, US, 11–20. https://doi.org/10.1145/2808194.2809449

[56] Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017).* Toulon, FR.

[57] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017).* Long Beach, US, 4077–4087.

[58] Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011).* Bellevue, US, 129–136.

[59] 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

[60] Niels van der Heijden, Helen Yannakoudakis, Pushkar Mishra, and Ekaterina Shutova. 2021. Multilingual and cross-lingual document classification: A meta-learning approach. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2021).* (Virtual Event), 1966–1976. https://doi.org/10.18653/v1/2021.eacl-main.168

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS 2017).* Long Beach, US, 5998–6008.

[62] Ricardo Vilalta, Christophe Giraud-Carrier, Pavel Brazdil, and Carlos Soares. 2011. Inductive transfer. In *Encyclopedia of Machine Learning*, Claude Sammut and Geoffrey I. Webb (Eds.). Springer, Heidelberg, DE, 545–548.

[63] 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.

[64] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018).* Melbourne, AU, 2321–2331.

[65] Ziyun Wang, Xuan Liu, Peiji Yang, Shixing Liu, and Zhisheng Wang. 2021. Cross-lingual text classification with heterogeneous graph neural network. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL 2021).* (Virtual Meeting), 612–620. https://doi.org/10.18653/v1/2021.acl-short.78

[66] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP 2020).* Online

event, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

[67] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2015)*. Denver, US, 1006–1011. https://doi.org/10.3115/v1/N15-1104

[68] Dejiao Zhang, Ramesh Nallapati, Henghui Zhu, Feng Nan, Cicero Nogueira dos Santos, Kathleen McKeown, and Bing Xiang. 2020. Margin-aware unsupervised domain adaptation for cross-lingual text labeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. (Virtual Event), 3527–3536. https://doi.org/10.18653/v1/2020.findings-emnlp.315

[69] Jing Zhang, Wanqing Li, Philip Ogunbona, and Dong Xu. 2019. Recent advances in transfer learning for cross-dataset visual recognition: A problem-oriented perspective. *Comput. Surveys* 52, 1, Article 7 (2019). https://doi.org/10.1145/3291124

[70] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. 2019. Bridging theory and algorithm for domain adaptation. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*. Long Beach, US, 7404–7413.