Active learning and negative evidence for language identification

Thomas Lippincott **Benjamin Van Durme**
Johns Hopkins University
{tom,vandurme}@cs.jhu.edu

Abstract

Language identification (LID), the task of determining the natural language of a given text, is an essential first step in most NLP pipelines. While generally a solved problem for documents of sufficient length and languages with ample training data, the proliferation of microblogs and other social media has made it increasingly common to encounter use-cases that *don't* satisfy these conditions. In these situations, the fundamental difficulty is the lack of, and cost of gathering, labeled data: unlike some annotation tasks, no single "expert" can quickly and reliably identify more than a handful of languages. This leads to a natural question: can we gain useful information when annotators are only able to *rule out* languages for a given document, rather than supply a positive label? What are the optimal choices for gathering and representing such *negative evidence* as a model is trained?

In this paper, we demonstrate that using negative evidence can improve the performance of a simple neural LID model. This improvement is sensitive to policies of how the evidence is represented in the loss function, and for deciding which annotators to employ given the instance and model state. We consider simple policies and report experimental results that indicate the optimal choices for this task. We conclude with a discussion of future work to determine if and how the results generalize to other classification tasks.

1 Background

Language identification (LID) is the task of classifying a document according to the natural language in which it is written (Lui and Baldwin, 2011). It is a special case of *text classification*, where a document is assigned a label l from a finite set of discrete values L. Such problems, and LID as a special case, have been widely studied for decades (Kranig, 2005; Jauhiainen et al., 2018), with recent state-of-the-art methods focusing on neural architectures over character representations (Joulin et al., 2017; Zhang et al., 2015). Most methods share the intuition (verified by many traditional studies) that the signal for LID comes from the character level. This intuition is reinforced by the difficulty that flexible neural architectures have unseating traditional n-gram methods (Lippincott et al., 2019): frequencies of short character-sequences seem to hold most signal for the task.

Negative evidence, for a feature or label, is explicit evidence that it is not present in or does not apply to an instance (Schneider, 2004): in this study, it refers to annotations that say "this document is *not* language X". A given annotator can only know a handful of languages, so in addition to the *positive evidence* when presented with one of them, there may be a much higher volume of implicit negative evidence, i.e. the documents whose language they couldn't recognize. Among text classification tasks, the LID task is a particularly acute, naturally-occurring example of this imbalance, in contrast to specific phenomena like linguistic structure or small-inventory tasks like named-entity recognition or sentiment, where annotators are expected to have a full grasp of the potential label-space.

Our use of model estimates to choose annotators has similarities to work on multi-armed contextual bandits (Riquelme et al., 2018), where the context includes both the new instance and the current model state. Similar to Bayesian last layer optimization (Weber et al., 2018) we focus on the final linear layer in the model, though rather than employing reinforcement learning we directly specify simple policies based on the output distributions. The choice of likeliest annotators is similar to Thompson sampling (Riquelme et al., 2018).

Figure 1[1] shows the performance of a LID model

[1] All figures in this paper show results for four values of *annotations_per_instance*, 2, 4, 8, and 16, indicated by

Proceedings of the 2nd Workshop on Data Science with Human in the Loop: Language Advances, pages 47–51

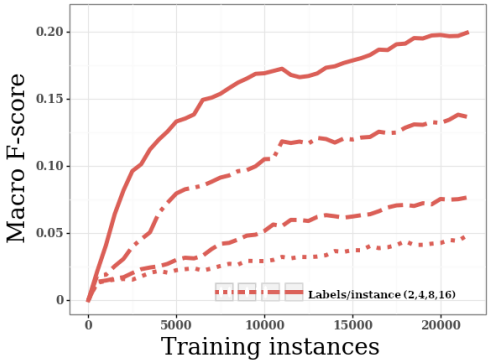

Figure 1: Performance when training **only** on negative evidence ("this is *not* language X") and no use of the model for routing instances to annotators or for estimating probabilities (i.e. the *Random* and *Uniform* policies described in Section 2). As languages are ruled out, probability-mass is shifted to the set containing the correct label. Improvements occur with more labeled instances (x-axis), and with more labels-per-instance.

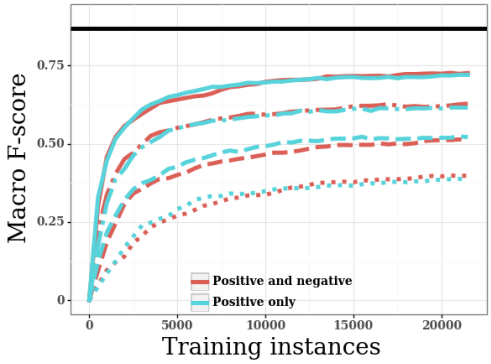

Figure 2: Performance using **positive** evidence, with and without negative evidence, also with no use of the model for routing or estimation. The signal identified in Figure 1 brings no advantages under this simple policy. Here and in subsequent figures, the black line indicates performance under the ideal situation where the full training set is positively labeled.

trained on just negative evidence. The curves clearly demonstrate that negative evidence contains useful information for the task. Figure 2 compares positive evidence with both positive *and* negative evidence, when negative evidence is incorporated naively (see descripions of *Uniform* and *Random* in Section 2). There is no performance gain from including the negative evidence under these conditions. Our goal is to preserve the signal from Figure 1, that is currenly being lost in Figure 2, and determine how to best employ annotation resources as training progresses.

2 Methods

Data The Twitter Language Identification dataset consists of 70k tweet IDs distributed evenly over 70 languages (Twitter, 2015). This provides a significant LID challenge due to the short, idiosyncratic messages and a large label space that includes many less-studied languages. We balance the data by randomly shuffling and keeping the first 400 instances of each language, and discarding seven languages with less than 400 total, as some tweets have become unavailable since the data set was published.

Model Our model truncates and pads input sentences to 128 characters, and maps each character into a randomly initialized 64-dimension embedding space. The $(128x64xBatchSize)$ tensors are fed to a two-dimensional CNN layer with filters of widths $w \in 1:5$ followed by ReLU nonlinearity. The CNN outputs are concatenated, fed to a dense linear layer and softmax to produce distributions over the labels. The choice of character-level CNNs of the given widths gives the model access to the same statistical information employed by traditional n-gram models.

Training We simulate L annotators, one per label, each only capable of recognizing their respective label. Starting with zero instances and a randomly-initialized model, at each step we are provided with 500 new instances, and for each of them, allowed to query $annotators\_per\_instance$ of the annotators, ranked according to an **annotator policy**. Each annotator returns *positive* if the instance is in the language they recognize, *negative* otherwise. This evidence is represented as a target categorical distribution for the model's loss function, according to a **representation policy**. The model is then trained via SGD for a maximum of 500 epochs, with learning rate=0.1, momentum=0.9, maximum iterations=500, minibatch size=32, early stop=20, learning rate reduction of magnitude=0.1 and patience=10, and dropout of 0.5 on the CNN outputs.[2] Note that we are training to convergence *between* receiving each new batch of annotated instances.

Evidence Given the annotations received during the training process, we can selectively employ evi-

increasingly-solid lines: for better readability, we omit this from the legends

[2]This training configuration produced optimal results on dev performance across all experimental conditions

dence: *Positive* and *Positive+Negative* are the most important point of comparison, while *Negative* was used in Figure 1 to illustrate the potential value of the negative annotations.

Annotation policies We experiment with two policies for ranking potential annotators for a new instance: in both cases, the instance is labeled by the first $annotators\_per\_instance$ annotators in the ranked list. The *Random* policy simply shuffles the annotators randomly. The *Likeliest* policy ranks annotators by the probability the model currently assigns to their language for the new instance, from most likely to least. The intuition is that getting annotations for the likeliest languages will either 1) correctly label the instance or 2) remove the maximal amount of misallocated probability mass under the current model parameters.

Representation policies We also experiment with two policies for representing negative evidence as an L-dimensional categorical distribution for input to the loss function (positive evidence is always the corresponding one-hot distribution). The *Uniform* policy, given negative evidence for labels L_{neg}, builds the distribution $P(l) = 0$ if $l \in L_{neg}$, otherwise $P(l) = \frac{1}{|L - L_{neg}|}$. The *Estimated* policy also sets $P(l) = 0$ if $l \in L_{neg}$, but otherwise sets $P(l)$ proportional to the model's current estimate of that language for the instance. We experimented with treating each potential label for each instance as a binary task via binary cross-entropy loss metrics, and with selectively propagating the loss depending on whether an annotation (positive or negative) had been seen for the task, but found that the best approach was to treat it as a categorical, with a KL-divergence loss metric.

```
train = []
for s in 1:S:
  new = get_more_instances(500)
  labeled = label(new, annotator_policy)
  train += encode(labeled, representation_policy)
  continue_training(model, train)
  recordScore(model, test)
```

Figure 3: Training procedure: the model is incrementally fed additional labeled documents, and each time SGD is run until dev set performance stops improving, at which point the test set score is recorded for the current amount of training data.

Measurements We perform five folds of each experiment, in which the full data set is randomly shuffled, and split into train/dev/test sets in 0.8, 0.1, and 0.1 proportions, respectively. Performance on the dev sets was used for early stopping and learning rate decay, while we report test performance averaged over the five folds. Variance was low, and we omit it from figures, but include it in Additional Materials. Figures show macro F1 score as a function of training instance count: line style corresponds to $annotators\_per\_instance$, while color corresponds to the policies being compared. The solid horizontal line at the top of the figures is a reference point of performance with complete, positive annotation of all instances.

We assume that we have access to one annotator per language and route each document as it arrives. In the absence of active learning, a document has a 1/70 chance to be labeled as its *correct* language, and 69/70 chance to be labeled as *not* one of the other languages. We also experiment with a simple policy of routing each document to the annotator for the model's current best-guess language, with the hypothesis this will reassign the most misallocated probability mass.

1. Treat the outputs as a categorical distribution by applying softmax and use standard cross-entropy between this and positive labels, making no use of negative labels. As extreme as this seems, it's not unrealistic in situations that focus on under-represented populations like sub-Saharan Africa or fine-grained distinctions like Arabic dialects to have an acute lack of on-demand expertise, or a large cost associated with it.

2. Treat the outputs and labels as independent Bernoulli distributions, and for each document only back-propagate the error from its explicitly-labeled language (be it a positive or negative label). This focuses the objective on the precise information the annotators have provided.

3. Treat the outputs as a categorical distribution, so identical to 1) for positive labels, but to additionally cast the negative labels as under-specified categorical distributions. With zero additional evidence we can use the maximum entropy distribution subject to the constraints of any negative labels, i.e. the uniform distribution with the negative labels set to zero and renormalized. We could also use external information, such as the population language distribution from another source, or an iterative prior estimation from current estimates, to

inform how the probability mass is distributed across the label space.

3 Results and Discussion

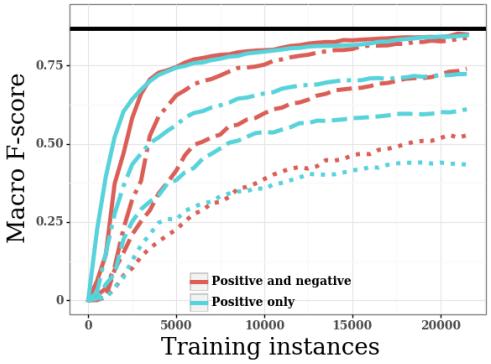

Figure 4: Comparison of the best approach using *Positive* evidence, and the best approach using *Positive+Negative* evidence, both of which use *Likeliest* annotation policy and *Estimated* representation policy. Given the same number of annotations-per-instance (line solidity), negative evidence provides significant performance improvements, after a brief initial delay (see Figure 6).

Figure 4 shows our primary result: the use of negative evidence, in combination with the *Estimated* representation policy and *Likeliest* annotator policy, produces large improvements over the best baseline approach using positive evidence alone. In particular, when the algorithm is given a small number of annotation opportunities per instance (2, 4, 8), it surpasses the baseline at 20k instances by 7, 15, and 10 points, respectively. Positive+negative with 4 annotations per instance exceeds performance of positive with 8 annotations per instance, and with 8 annotations per instance is within 3 points of performance with perfect positive evidence.

On the other hand, the *Positive+Negative* models show performance delays compared to the corresponding positive models. Because the negative instances rely on the quality of the current estimates for constructing target distributions, these are initially quite poor until the model escapes its random initialization. This escape is easier to accomplish with more annotations, which can be seen by comparing where the *Positive+Negative* models start outperforming their *Positive* counterparts.

Figures 6 and 5 compare representation and annotator policies under the simplest configurations. The delayed performance is clearest between the

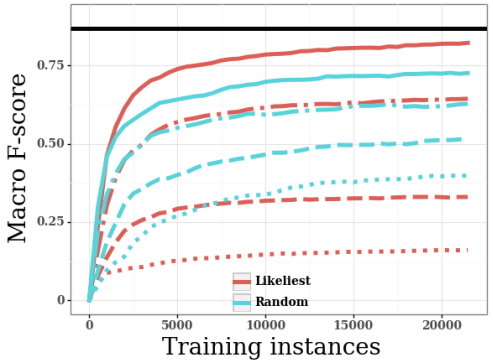

Figure 5: Comparison of *Random* and *Likeliest* annotator policies, both using the *Uniform* representation policy and *Positive+Negative* evidence. The *Likeliest* policy lacks the performance delay seen in Figure 4, but also falls short of the improvement given more training instances.

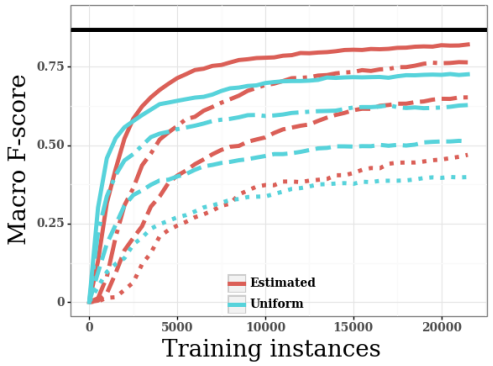

Figure 6: Comparison of *Uniform* and *Estimated* representation policies, both using the *Random* annotator policy and *Positive+Negative* evidence. The *Estimated* policy exhibits the performance delay of Figure 4, indicating this stems from the model's initial poor estimates of unseen label probabilities, but also shows the value of those estimates in the improvements once those estimates are based on sufficient training instances.

representation policies. The *Estimated* representation policy also provides the most performance improvement, eventually providing benefits for every value of $annotators\_per\_instance$. The annotator policies, on the other hand, only provide significant benefits when $annotators\_per\_instance \geq 8$. Referring back to Figure 4 confirms the combination of both policies outperforms either in isolation.

4 Conclusions and future work

We have demonstrated a general method for exploiting negative evidence using an underlying virtuous cycle, where an improved model leads to better annotator selections and more accurate target distri-

butions. These appear to be crucial ingredients for negative evidence to provide benefit: when we remove them, and always select annotators at random and employ CMEDs, the positive+negative models provide no advantage over the positive-only models throughout training. It may be that the benefits would only materialize on larger training sets, but since the models already approach the optimum (black line) this is never seen in practice for this LID task.

Negative evidence can be gathered (and simulated) for any classification task, and it is an open question whether the same approaches will generalize. In particular, we would like to run experiments in speech processing, which has its own language identification problem as well as a number of tasks with even sparser annotation abilities (e.g. speaker identification), and image recognition.

The initial performance delays we observed in Figures 4 and 6 come from the *Estimated* policy's use of the model before it has sufficiently improved. Representation policies that take this into account should be able to shift the early performance curves to the left, similar to work on multi-armed bandits (Weber et al., 2018) emphasizing the importance of uncertainty estimates. The annotation policies we considered can only gather annotations for incoming instances: it may be useful to expand this to include *all* currently-known instances, to allow more flexible shifting of probability mass using the same amount of annotation resources.

References

Tommi Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. 2018. Automatic Language Identification in Texts: A Survey. *CoRR*, abs/1804.08186.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Simon Kranig. 2005. Evaluation of language identification methods. *Bakalárska práca, Universität Tübingen, Nemecko*.

Tom Lippincott, Pamela Shapiro, Kevin Duh, and Paul McNamee. 2019. JHU system description for the MADAR Arabic dialect identification shared task. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 264–268, Florence, Italy. Association for Computational Linguistics.

Marco Lui and Timothy Baldwin. 2011. Cross-domain feature selection for language identification. In *Proceedings of 5th international joint conference on natural language processing*, pages 553–561.

Carlos Riquelme, George Tucker, and Jasper Snoek. 2018. Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. In *International Conference on Learning Representations*.

Karl-Michael Schneider. 2004. On word frequency information and negative evidence in Naive Bayes text classification. In *Advances in Natural Language Processing*, pages 474–485. Springer.

Twitter. 2015. Evaluating language identification performance.

Noah Weber, Janez Starc, Arpit Mittal, Roi Blanco, and Lluís Màrquez. 2018. Optimizing over a Bayesian Last Layer. In *Advances in neural information processing systems*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.