Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 3520–3527
Marseille, 11–16 May 2020

Search Query Language Identification Using Weak Labeling

Ritiz Tambi, Ajinkya Kale, Tracy Holloway King
Adobe Inc.
San Jose, California, USA
{tambi, akale, tking}@adobe.com

Abstract

Language identification is a well-known task for natural language documents. In this paper we explore search query language identification which is usually the first task before any other query understanding. Without loss of generalization, we run our experiments on the Adobe Stock search engine. Even though the domain is relatively generic because Adobe Stock queries cover a broad range of objects and concepts, out-of-the-box language identifiers do not perform well due to the extremely short text found in queries. Unlike other well-studied supervised approaches for this task, we examine a practical approach for the cold start problem for automatically getting large-scale query-language pairs for training. We describe the process of creating weak-labeled training data and then human-annotated evaluation data for the search query language identification task. The effectiveness of this technique is demonstrated by training a gradient boosting model for language classification given a query. We out-perform the open domain text model baselines by a large margin.
Keywords: language detection, language identification, search queries, short text, weak labeling

1. Introduction

Language identification is key to accurately applying deeper natural language processing in contexts where the language of the text is not known. Relatedly, it can be used to classify texts by language and hence ensure that a set of texts are all in the same language. In this paper we focus on search queries[1] for Adobe Stock. Adobe Stock allows users to search over a set of images for purchase. The searches can use keywords, images, or a combination thereof; for language identification we are only concerned with the keywords. These keywords reflect vocabulary used to describe items in the world (e.g. *dog*, *woman*, *coffee cup*), abstract concepts (e.g. *happiness*, *success*), and some domain specific vocabulary (e.g. *icon*, *vector*, *background*). We want to know the language of the query in order to apply processes like named entity recognition and dependency parsing to improve query understanding and hence search result quality. In addition, we want to provide auto-complete suggestions and related searches in the correct language, i.e. the language of the user's issued queries.

Language identification systems are generally trained on well-edited text. They use a variety of signals including: known words and especially known closed class words (e.g. determiners, auxiliaries); character n-grams (often trigrams); punctuation and spaces, which can be viewed as a subset of characters; upper vs. lower casing, which can be viewed as a subset of characters. These general purpose language identification models work well for longer texts which have similar editing style. However, their performance drops significantly on shorter texts such as search queries and shorter tweets (Cavnar and Trenkle, 1994; Sibun and Reynar, 1996; Baldwin and Lui, 2010), as do many NLP components (Baldwin et al., 2013).

There are two reasons for this. The first is simply that the shorter text provides less signal, especially for discriminating between similar languages (e.g. between the Scandinavian languages). However, the degradation is often much greater than expected. This is because of the second reason, namely that short text is often missing many of the signals that are seen in the training text (Baldwin et al., 2013; Wang et al., 2014; Sun et al., 2016). In search queries, capitalization is meaningless, there are frequent misspellings, punctuation is missing or used differently, closed class words are dropped, and spacing and punctuation often reflect the native language (keyboard) of the user and not the language of the words they are using. For example, unlike in long text, in our Adobe Stock search queries there are some prepositions (e.g. *boy on beach*) but almost no determiners or auxiliaries, and verbs are generally progressive participles (e.g. *man sitting at computer*). This is similar to short text more generally and especially to web and e-commerce search queries. Additionally, in tweets and certain search domains, specialized vocabulary (e.g. hashtags, model numbers, and brand names) often have character n-grams that are more in line with other languages.

For the application which we were interested in, we did not need to identify transliterated languages, e.g. identify Hindi when written in the Latin alphabet. There are types of short text, including certain search query applications, where transliterated text is common and being able to identify these is crucial. However, in our context the character set used is an extremely strong indicator of the language. Our domain has some code switched queries in which a single query has multiple languages. This is particularly noticeable for Japanese-English combinations. We leave language identification for code switched queries (Zhang et al., 2018) to future work.

The issue of having less signal due to the shorter text can be partially overcome by limiting the languages to be detected to the ones that are most likely in the application and by further providing input on the likely language. The likely language can be influenced by the language of the site (e.g. if the search page is in Japanese, the query is likely to be in Japanese), the location of the user when known, and the most likely language overall (e.g. in our context English queries are common on all the sites even when En-

[1] Since these queries are written text, we do not explore methods for language identification for short utterances (Fernando et al., 2017; Bekker et al., 2016; Travadi et al., 2014).

glish is not an official language of the country). In our applications, we only want to identify languages where we can use the language identification to improve the customer experience, e.g. by applying the correct deeper natural language processing or by limiting auto-complete suggestions to the correct language. This means that identifying constructed languages like Esperanto and Klingon is not necessary or useful. Identifying languages like Latvian and Swahili, which are currently not supported on Adobe Stock, is only useful for understanding what languages to consider expanding the experience for. So, by default we want the language identifier to focus on a handful (<20; currently 8) languages, where English will be the overall default.

We tried two language identifiers out of the box.[2] One of the language detectors was trained on an extremely large number of languages, including differentiating different versions of the same language (e.g. Chinese as spoken in China and Taiwan, French as spoken in France and Canada) and constructed languages (e.g. Esperanto, Klingon). The short nature of the queries given the large number of target languages caused issues for accuracy and highlighted issues which, upon closer inspection, affected other systems. Many language identifiers use closed class words such as determiners and prepositions and the character ngrams within them as heavily weighted features. However, since queries are very short and generally drop closed class words, these features cannot be reliably applied.

This paper describes the process of automatically creating a large-sized, weak-labeled training set and then manually creating a smaller, human-annotated evaluation set used to train and evaluate our search query language identification model. The training data was created by taking a seed dictionary of known language terms and finding their nearest-neighbors in the query logs. We then found the nearest-neighbors to that larger set of labeled query terms and used a voting schema to determine their language. The result is a ∼664K query set with weak labels for language. We then trained a CatBoost language identification model. To evaluate the model, we manually annotated ∼65K queries through a crowd-sourcing task. To ensure accuracy, each language was its own task with the instructions provided only in that language and we mixed in queries from other languages, focusing on English and on languages likely to be confused with the target language.

Key contributions of this work include:

- Validation of the hypothesis that out-of-the-box, state-of-the-art language identifiers do not work optimally on short text such as search queries;
- Validation of the hypothesis that noisy, weakly labeled training data, which is available at scale with no manual annotations, result in a high-accuracy language identifier for short text;
- A simple but effective two-stage label creation and propagation strategy which takes advantage of off-the-shelf components such as a seed word list, combined with in-domain embeddings model for nearest neighbor queries;
- Test set creation via human annotation using model-based bootstrapping and language interleaving.

Rest of the paper is organized as follows - we first describe how we trained our model by bootstrapping from a seed dictionary and large list of naturally occurring search queries, resulting in a large-scale, weakly-labeled training set (section 2.). We then discuss the evaluation of the model and hence indirectly the quality of the training data, including the creation of a gold-standard human-annotated evaluation set (section 3.). We end with next steps (section 4.).

2. Language Identification Model Training

Our goal was to train a multi-class prediction model for seven languages and an "other" (i.e. unknown) language on search queries: English, German, Japanese, French, Italian, Spanish, Portuguese, and the other/unknown class.[3]

2.1. Training Data Creation: Weak Labeling

Our first task was to create training data. The goal was to obtain a large number (hundreds of thousands) of naturally occurring, in-domain search queries with labels as to their language. Human labeling that many queries would have been prohibitively expensive and time consuming. Instead, we employed a weak labeling technique that allows us to scale to such a large number of training samples automatically.

We paid particular attention to how we sampled the queries for the weak labeling (this section) and the evaluation (section 3.1.) tasks for two reasons. First, the preponderance of queries were in English due to significantly greater query volume on the US site and to the use of English language queries on other sites. Second, we wanted to ensure sampling across frequent (head) and infrequent (tail) queries.

2.1.1. Seed Dictionary

We started with a seed dictionary of words whose language is known. We then used this set of known-language words to weakly label a large number of search queries from the Adobe Stock query logs with their most-likely language. We labeled the queries with the language in two steps: we first created labels on queries close to the seed terms and then propagated them to similar queries to get enough training samples from a limited size seed dictionary.

The seed dictionary was compiled from the MUSE dictionaries (Conneau et al., 2018). The MUSE dictionaries were originally compiled for creating multilingual word embeddings. The number of seed words for the seven languages we are interested in are shown in the middle column of table 1.

2.1.2. Label Creation

The seed words were used to find a nearest neighbor set of queries using a custom-trained in-domain KNN model (see section 2.1.4.). The nearest neighbors to queries which corresponded to the seed dictionary words were annotated

[2] We also applied some queries to Google translates auto-language detection. We did not conduct a systematic evaluation with respect to Google due to throttling limits on their side.

[3] Our model also predicts Korean, but since Korean was identified extremely accurately through its character set, we do not include results on Korean here.

with the seed word language if they were above a threshold. In some cases the number of labeled queries is less than the number of seeds (e.g. English, Portuguese) because many of the seed words have no equivalent in the query logs. In other cases the number of labeled queries is greater than the number of seeds (e.g. German, Japanese) when more than one query was close to the seed. This first step of the training data creation process is called the Label Creation task. The number of queries labeled in this initial label creation task are shown in the last column of table 1.

Language	Seeds	Labeled Queries
English	92,843 (37.7%)	52,777 (28.1%)
German	27,892 (11.3%)	42,735 (22.7%)
Japanese	20,975 (8.5%)	25,652 (13.6%)
French	23,233 (9.4%)	23,054 (12.3%)
Italian	26,164 (10.6%)	14,949 (8.0%)
Spanish	34,425 (14.0%)	21,312 (11.3%)
Portuguese	21,023 (8.5%)	7,551 (4.0%)
Total	246,555 (100%)	188,030 (100%)

Table 1: Number and distribution of MUSE seed words and their corresponding nearest neighbors in the query logs

2.1.3. Label Propagation

Once we had the initial labeled queries from the Label Creation task, the remaining queries were annotated similarly by finding their nearest neighbors in the set of queries sampled from the query logs. This query pool was chosen by stratified sampling of queries over a 1-year period. The stratified sampling ensured representation of head, torso, and tail queries across languages. For each query, from its nearest neighbor set, a subset of queries with cosine similarity above a specified threshold were selected. If the number of queries in that subset crossed a threshold length, a majority vote was taken to annotate the query. This second round of query labeling is referred to as the Label Propagation task.

The process is shown schematically in figure 1 with the Label Creation task feeding the Label Propagation task.

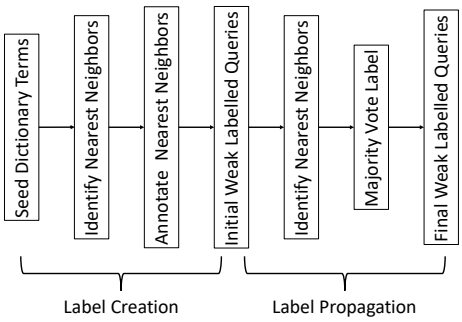

Figure 1: Label creation and propagation process

The Label Creation and Label Propagation processes provided ∼664,000 annotated queries for training a model (table 2). The length (i.e. number of words) of the queries reflects that of the overall search queries, where 2-word queries are by far the most frequent (for details see the bottom rows of table 7). We used 90% of this data for model training, with 5% reserved for tuning and 5% for evaluation. This evaluation data was extremely noisy due to the way in which it was constructed and so we also created a human-annotated evaluation set (section 3.). We estimate that the noise in this training data was over 10% for most languages based on running our final trained model over the evaluation portion of the training and comparing this to the human-annotated results (see table 6).

Language	Queries
English	124,472 (18.7%)
German	122,951 (18.5%)
Japanese	56,707 (8.5%)
French	62,787 (9.5%)
Italian	33,332 (5.0%)
Spanish	53,878 (8.1%)
Portuguese	12,400 (1.9%)
Other	197,744 (29.8%)
Total	664,271 (100%)

Table 2: Distribution of language labels in the training data

2.1.4. Query Embeddings & Nearest Neighbor Search

For training the in-domain (i.e. Adobe search query) KNN model used to find the nearest neighbors in the labeling process (sections 2.1.2. and 2.1.3.), we had to obtain optimal, in-domain query embeddings. That is, we needed embeddings which reflected the Adobe Stock search queries as opposed to general, off-the-shelf vocabulary such as that found in Wikipedia.

We trained a fastText skipgram model (Bojanowski et al., 2016) on ∼6 million Adobe Stock queries. This provided embeddings optimized to the shorter context found in search queries. Multiple fastText models were trained on a variety of parameters. The optimal model was found by a qualitative (cosine) similarity analysis comparing the nearest neighbor queries found by the different fastText parameters. Since our label propagation job heavily relied on a fast nearest neighbor search, we trained an approximate nearest neighbor model using Faiss (Johnson et al., 2017). A separate Faiss index was trained for each of the query embedding sets obtained from the different fastText models. We explored the following fastText parameters:

- Embedding dimensions: 100, 300
- Minimum character subword size (n): 1, 2, 3
- Maximum n: 4, 5, 6

Using lower min-n values led to higher quality, i.e. more semantically similar, neighbors, especially for Japanese and Korean. With higher min-n, Japanese and Korean had more code-switched results (i.e. queries which contained both Japanese and English words) and some queries even had nearest neighbors entirely from other languages such as Russian. So, we chose a min-n of 1. Using a high max-n (e.g. a max-n of 6) with a min-n of 1 led to irrelevant results for

Latin script languages. So, we chose a max-n of 4. The qualitative analysis of the various combinations of min-n and max-n confirmed that the best result for converting the textual queries to sentence vectors comes from a min-n of 1 and a max-n of 4.[4] A similar analysis of values for the feature dimension size showed no qualitative difference between using the smaller 100 dimension vectors or the larger 300 dimension ones. We hypothesize that this is due to the short length of the queries (table 7). As a result, we chose the more compact 100 dimension vectors.

2.2. Model Training

We trained three GBT models: Xgboost, CatBoost and LightGBM.

Since even state-of-the-art language identifiers sometimes predict languages which do not match the character set (e.g. predicting Japanese for a text in Latin characters), in addition to the query vectors described in section 2.1.4., we provided additional parameters to the model that represent the fraction of character set per language. To create these, for each language we determine the character set. For example, for English the character set is the 26 letters of the lower-cased alphabet and 10 digits.[5] For each query, the percentage of the characters in the query from each language is computed. For example, the Japanese query:

ジンボはリンゴを食べる

would have English-fraction: 0, German-fraction: 0, ... Japanese-fraction: 1.0.

Each query is converted to a feature vector of length 109, comprising a 100 dimension fastText vector and 9 language-fractions per language.

We trained the model only on Adobe Stock queries and similarly evaluated against a test set of Adobe Stock queries. Nothing in the techniques described here is domain or language specific, although we discuss some anticipated challenges for language identification for Adobe help queries in section 4.

Of the three systems we trained, CatBoost was the clear winner. In the remainder of this paper, we only report results for the CatBoost model.

3. Evaluation

In this section, we discuss the test set creation (section 3.1.), introduce the baseline systems (section 3.2.), present the results (section 3.3.), and provide error analysis (section 3.4.).

3.1. Query Language Test Set Creation

Since the training data generated by the Label Creation and Label Propagation process consisted of weak labels, a golden test set was created via a crowd sourced task.[6] We created a test set for each language the system would predict (English, German, Japanese, French, Italian, Spanish, Portuguese).

To create the query set to be labeled by the human annotators, raw queries are passed to the CatBoost model. This generated language labels and prediction scores for each query. Using the generated labels, we selected 10,000 queries for each predicted language for manual annotation. Since these were predicted labels, the actual language distribution of each set varied.

We supplemented the 10,000 queries for each language with 1,000–2,000 queries from other languages to ensure that the human judges were on task. English was mixed in with all the languages since it is used on all locales and all the human annotators knew some English. Korean was mixed in with Japanese to ensure the annotators were not simply judging based on a non-Latin character set. Portuguese and Spanish were mixed with each other since the language identifiers frequently confused them. The language of these mixed-in queries was determined by the labels predicted by the model and so was not always accurate. Only queries where the model probability was $>.99$ were selected. With this threshold, initial inspection by Adobe-internal language experts indicated that the labels were accurate enough to ensure a mix of languages despite the fact that there were some errors. This was confirmed empirically by the subsequent human labeling.

The targeted mix for each language test set is shown in Table 3. All sets aimed for 10,000 queries in the targeted language.

Target	**10,000 Core Queries**	**Mixed-in Queries**
English	90% English	10% unknown[7]
German	80% German	20% English
French	80% French	20% English
Japanese	80% Japanese	10% English
		10% Korean
Spanish	80% Spanish	10% English
		10% Portuguese
Portuguese	80% Portuguese	10% English
		10% Spanish
Italian	80% Italian	20% English

Table 3: Language mix (targeted based on predicted labels) used for the manual annotation task

To further ensure the quality of the human annotation, we used Adobe-internal language experts to label $\sim$1000 queries for each language to use as test queries ("honeypots") within the task. Running the task first internally also served to fine-tune the annotation guidelines, especially around how to handle digits and nonsense queries. To further ensure that only annotators who knew the language in question did the task, we posted the instructions for each language's task only in that language. This way only annotators who could read the language in question could understand and complete the task.

[4] In future work, we plan to investigate the impact of contextual embeddings as a solution to vector averaging based sentence embeddings using word vectors.

[5] Accented characters such as the *é* in *café* were not included in English. Similarly for other languages the character set focused on the core characters.

[6] We used Figure8 (Figure Eight Inc., 2019) for this task in order to have access to non-English speakers and improved quality control.

[7] Unknown comprises languages other than English that used Latin script, including languages not among the seven identified by the model.

3.2. Baseline Systems

We compare our CatBoost model trained on the weak-label data to three baseline systems. These comprise two publicly available machine learned models and one heuristic method.

- fastText (Joulin et al., 2016; Bojanowski et al., 2016; Grave, 2017): based on subword features used in the fastText classification models.
- langid.py (Lui and Baldwin, 2012): trained over a naive Bayes classifier with a multinomial event model (McCallum and Nigam, 1998), over a mixture of byte n-grams ($1 \leq n \leq 4$), selecting features which are sensitive to language but not domain (Lui and Baldwin, 2011).
- Heuristic: Assume the language of the locale (site url) for the search (a variant of majority wins).

The two publicly available, off-the-shelf systems are trained on standard text, not on query text. So, even though the Adobe Stock queries use standard vocabulary, especially nouns and adjectives, the fact that they are very short and have relatively few determiners, prepositions, and other closed class words predicts that these systems will not perform well. The langid.py (Lui and Baldwin, 2012) language detection model was chosen because it was reported to perform comparably to other language detection systems for long text (i.e. LangDetect, TextCat, and CLD) and better than systems which were specifically designed for short text (Hammarstrom, 2007; Ceylan and Kim, 2009; Vatanen et al., 2010; Carter et al., 2013). However, the short text they were evaluating against was Twitter and tweets are significantly longer than our search queries. The fastText (Joulin et al., 2016; Grave, 2017) language detection model was chosen because it reports to be more accurate and faster than the langid.py model. It uses the subword features generally used in fastText classification models and then compresses the model through feature selection and weight quantization. In contrast to the off-the-shelf models, we expected that the baseline of assuming the language of the query is the same as the locale will be a difficult baseline to beat, especially in English-speaking locales. In non-English-speaking locales, English queries often occur. In fact, this mix of languages is the main practical motivation for creating a query language detector. In our query logs, which included queries issued on sites which are not in one of the seven languages we are interested in detecting, 91% of unique queries (types) occur on only one site, but the more frequent queries appear on many sites with the result that 39% of query impressions (tokens) appear on only one site.[8] In contrast, 1% of unique queries appear on 5 or more sites, which corresponds to 37% of query impressions. To understand why locale in general is not enough to determine query language, consider the queries in table 4. The third column of the table indicates how many sites the query was issued on, while the fourth column indicates how many of those sites had a locale which matched the language of the query (e.g. a German query on a German or Austrian site; an English query on a US, UK, Canadian, or Australian site).

[8] We are unable to reveal the raw numbers for proprietary reasons.

Query	Language	# Sites Issued	# Matching Locale-Lang.
niños	Spanish	5	2
weihnachten	German	10	2
teamwork	English	11	2
team work	English	9	4
social media	English	15	4

Table 4: Example queries with how many sites they were issued and how many of those sites matched the language of the query

3.3. Results against the Gold Standard

Table 5 shows the comparison between our domain-specific model, the two generic pre-trained models, and the simple baseline of assuming that the query language is the same as that of the locale. In all cases, the query-specific model outperforms the pre-trained models in precision, recall, and F1 score. The results of the fastText model are particularly interesting in that precision was significantly higher than recall for all languages except Japanese where the recall was perfect (i.e. it identified all the Japanese queries as Japanese) but the precision was very bad (i.e. it frequently identified non-Japanese queries as Japanese).

When training our CatBoost model (section 2.2.) we did not have a manually-annotated evaluation set available to tune against. Instead, we split the weak labeled queries (section 2.1.) into the standard train-test-eval set and used the weak labels to determine the parameter settings. The precision, recall, and F1 results for our model against the weak labeled data are shown in table 6. These are much worse than the results against the true, gold standard data in table 5. This reflects the fact that the weak labeling process is far from 100% accurate and hence the final model is doing a better job of predicting language than the weak labeling process. This hypothesis was confirmed by manually spot-checking the weak label test data shown in table 6 when the prediction was "incorrect" and confirming that the majority of the time this was due to incorrect label propagation. It is a strength of this approach that with sufficient quantities of weakly labeled data, the entire model training process, including parameter tuning, can be conducted on weakly labeled data. The human-annotated data is only needed to confirm the exact performance of the system and hence to determine whether the system is accurate enough to use in applications.

For the heuristic baseline, our hypothesis that using the language of the site (i.e. locale) would be a hard baseline to beat proved incorrect. For all seven languages, the CatBoost model outperformed the heuristic, often by a significant margin. Only Japanese was close with the heuristic F1 score of 0.95 compared to the CatBoost F1 score of 0.99. This low performance confirms the need for language identification of queries since users are frequently issuing queries whose language does not match that of the site they are on. For the gold standard evaluation queries, $\sim$7% of the queries were issued on sites where the language of the locale was not one of the seven languages, e.g. the queries *christmas business* and *money family* were issued on the Finnish site and the queries *soup milk herbs* and *metal cup* on the Russian site.

Language	**CatBoost Query**			**langid.py** (generic)			**fastText** (generic)			**Heuristic** (locale)		
	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
English	0.95	0.96	0.96	0.80	0.68	0.73	0.98	0.38	0.55	0.88	0.70	0.79
German	0.98	0.97	0.98	0.78	0.70	0.74	0.88	0.09	0.17	0.64	0.93	0.79
Japanese	1.00	0.99	0.99	1.00	0.80	0.89	0.31	1.00	0.47	0.93	0.97	0.95
Spanish	0.91	0.96	0.94	0.64	0.56	0.60	0.90	0.41	0.57	0.59	0.62	0.61
French	0.96	0.97	0.97	0.77	0.65	0.70	0.95	0.39	0.55	0.62	0.82	0.72
Italian	0.98	0.93	0.95	0.77	0.67	0.72	0.90	0.42	0.57	0.68	0.72	0.70
Portuguese	0.98	0.92	0.95	0.86	0.65	0.74	0.77	0.57	0.66	0.75	0.52	0.63

Table 5: Result comparison between custom-trained CatBoost query model, the generic langid.py and fastText pre-trained models, and the heuristic baseline of using locale to predict language

Language	Precision	Recall	F1
English	0.90	0.78	0.83
German	0.90	0.94	0.92
Japanese	0.96	0.97	0.97
Spanish	0.81	0.89	0.85
French	0.89	0.92	0.90
Italian	0.78	0.83	0.81
Portuguese	0.73	0.41	0.53

Table 6: Weak label evaluation set with CatBoost model

3.4. Error Analysis

In general, we expect languages which are closely related to one another to be more likely to be confused with one another. In addition, given that we use the language character set as a feature, we expect vanishingly few errors where the language is identified as one with the wrong character set. These hypotheses are borne out by the confusion matrix in figure 2.

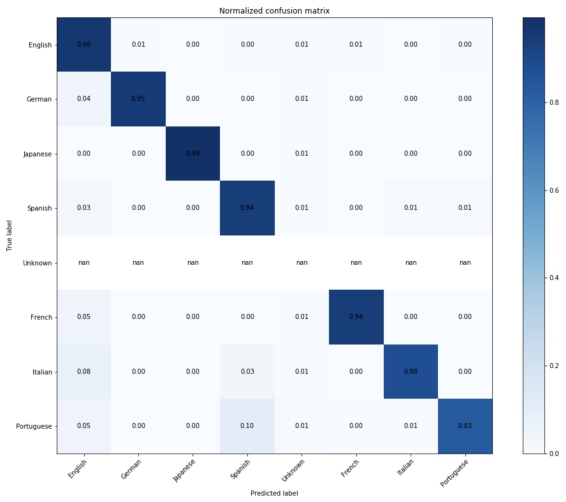

Figure 2: Confusion matrix for CatBoost model against gold test data

Portuguese and Spanish are the hardest two languages to distinguish from one another, with Portuguese queries generally being misidentified as Spanish. For example, *letras giz vetores* (Spanish) was predicted to be Portuguese and *cacau seco* (Portuguese) as Spanish. Japanese, in contrast, is basically never misidentified since its character set is unique among the languages. However, one counter example to this is when the Latin alphabet characters used were associated with Japanese double-wide characters, e.g.

ｂｏｘｉｎｇ　ｆｅｍａｌｅ

(English) was predicted to be Japanese. In addition, although predictions of Japanese were always correct (i.e. high precision), some clearly Japanese queries were predicted as the unknown/other class, suggesting that our features for language-fraction are not strong enough in some cases (section 2.2.).

Queries which contain digits as well as words sometimes fooled the model, e.g. *10 jahre kerzen* (German) was predicted as English, *24 y 31* (Spanish) as Korean, *50 anni compleanno* (Italian) as English. As future work, the training data needs to be further supplemented with more mixed digit-word queries or the digits need to be masked when inferencing.

Interestingly, all the Latin alphabet languages are periodically misidentified as English. This appears to be due to many borrowed English words being used, especially in this search application. English phrases appear surprisingly frequently in search queries on non-English language Adobe Stock sites.

As expected, query length has an effect on accuracy, with shorter queries having lower F1 scores. This was particularly marked for identifying 1-word English queries. Fortunately, and somewhat unintuitively, 1-word queries are less popular than 2-word queries in Adobe Stock. In addition, on English language sites, English is a good estimate for a default. A breakdown of F1 score by query length for the seven languages is shown in table 7.

	F1 Score			
Language	**1 word**	**2 word**	**3 word**	**4+ word**
English	**0.73**	0.94	0.98	0.99
German	0.96	0.98	0.98	0.99
Japanese	0.99	1.00	1.00	0.99
Spanish	0.85	0.94	0.96	0.95
French	0.90	0.96	0.98	0.98
Italian	0.89	0.95	0.97	0.96
Portuguese	0.89	0.94	0.97	0.97
# Queries	10,979	27,391	18,296	6.041
% Queries	17.5%	43.7%	29.2%	9.6%

Table 7: F1 Score by query length for each language. Note the drop for English 1-word queries.

3.4.1. Error Analysis for the Pre-trained Models

For the pre-trained models, the most surprising issue arose with languages which were predicted even though the character set used in the words did not match that of the predicted

language. This was particularly striking for predictions of Chinese and Japanese. Upon closer inspection, two factors heavily influenced the prediction of these languages even though in the search query context they were irrelevant.

The first was that the space character used in Chinese, Japanese, and Korean (CJK) is a different character than that used for languages traditionally written with the Latin alphabet. The presence of a single CJK space even when combined with all non-CJK character words was enough to fool the system. We hypothesize that search queries disproportionally have this mix because users are switching between languages, including cutting-and-pasting from the web, and so end up with mixed space-word character sets for mono-lingual queries.

The second factor was that Japanese queries sometimes had a backspace as the initial character. This character was not normally visible but even when combined with all non-Japanese character words the system would predict Japanese. We hypothesize that the Japanese (and possibly many CJK) methods for entering characters can result in these initial backspaces.

A third lesser factor was that one of the systems predicted CJK when the query was in all upper case Latin letters and one of those letters had a diacritic on it (e.g. *MANOS EN PUÑOS* predicted as Korean, *BRUNE COUPE CARRÉ* predicted as Japanese). Since capitalization has no signal in search queries, unlike in edited longer text, having a language identifier which is sensitive to capitalization is a major drawback.

4. Conclusions and Future Work

Off-the-shelf language identification models are inaccurate for domain-specific search queries, e.g. in our Adobe Stock search domain, for several reasons: the queries are short (1–5 words); they do not contain many closed class words; we want to identify just seven languages (English, German, Japanese, French, Italian, Spanish, Portuguese) with high confidence. A key contribution of this paper is validation of the hypothesis that noisy, weakly labeled training data, which is available at scale and which requires no manual annotation, results in a high-accuracy language identifier for short text such as search queries. In order to train a domain-specific language identification model, we bootstrapped domain-specific training data in a two-step process. We first used a seed dictionary from MUSE to find nearest neighbor queries for the seven languages using a KNN model trained on the Adobe Stock query domain. We then propagated the labels from those queries to their nearest neighbor queries from a stratified sample of 1 year of in-domain queries. We trained a CatBoost model using 100 dimension fastText vectors and 9 language-fraction features. We used crowd-sourcing to create a domain-specific evaluation set focused on the seven languages of interest. To ensure the attention of the human annotators we included 10-20% of other language queries in each task and we supplied task instructions only in the target language. Overall f-score on the final model was 0.98 on the test set and inference latency is $\sim$3ms on our standard CPU machines used for query processing, low enough for query-time production use.

We have two areas for future research. The first is to modify the system to allow language identification of queries with Adobe product names. These do not occur frequently in the Adobe Stock queries, but they are extremely frequent in queries for other Adobe products. The same product name (e.g. Photoshop, Premier Pro) is used regardless of the language. When a query consists of only a product name, its language is indeterminate; that is, the query *premiere pro* could be English, German, Japanese, etc. When a query consists of a product name and other words, those words determine the language, e.g. *reinstall photoshop* is an English query. The second area of future work is handling code switched queries. We plan to build on (Zhang et al., 2018)'s analysis of code switching in short text. As in their scenario, in our Adobe search use cases only certain language pairs are commonly code switched, e.g. English and Japanese occur frequently together, and so the problem can be simplified to detecting those switches.

5. Acknowledgements

We would like to thank Alex Filipkowski and Kate Sousa for their expert advice and help in running the manual annotation task.

6. Bibliographical References

Baldwin, T. and Lui, M. (2010). Language identification: The long and the short of the matter. In Proceedings of NAACL HLT 2010, pages 229–237.

Baldwin, T., Cook, P., Lui, M., MacKinlay, A., and Wang, L. (2013). How noisy social media text, how diffrnt social media sources? In International Joint Conference on Natural Language Processing, pages 356–364.

Bekker, A. J., Opher, I., Lapidot, I., and Goldberger, J. (2016). Intra-cluster training strategy for deep learning with applications to language identification. In IEEE International Workshop on Machine Learning for Signal Processing.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. In CoRR.

Carter, S., Weerk, W., and Tsagkias, M. (2013). Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation*, pages 195–215.

Cavnar, W. B. and Trenkle, J. M. (1994). Ngram-based text categorization. In Proceedings of the Third Symposium on Document Analysis and Information Retrieval.

Ceylan, H. and Kim, Y. (2009). Language identification of search engine queries. In Proceedings of ACL2009, pages 1066–1074.

Conneau, A., Lample, G., Denoyer, L., Ranzato, M., and Jégou, H. (2018). Word translation without parallel data. In ICLR.

Fernando, S., Sethu, V., Ambikairajah, E., and Epps, J. (2017). Bidirectional modelling for short duration language identification. In INTERSPEECH.

Figure Eight Inc. (2019). Data annotation platform. `https://www.figure-eight.com/`.

Grave, E. (2017). Language identification. Blog post at https://fasttext.cc/blog/2017/10/02/blog-post.html.

Hammarstrom, H. (2007). A fine-grained model for language identification. In Proceedings of iNEWS07, pages 14–20.

Johnson, J., Douze, M., and Jégou, H. (2017). Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Lui, M. and Baldwin, T. (2011). Cross-domain feature selection for language identification. In Proceedings of 5th International Joint Conference on Natural Language Processing, pages 553–561.

Lui, M. and Baldwin, T. (2012). Langid.py: An off-the-shelf language identification tool. In Proceedings of the ACL 2012 System Demonstrations, ACL '12, pages 25–30. Association for Computational Linguistics.

McCallum, A. and Nigam, K. (1998). A comparison of event models for Naive Bayes text classification. In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization.

Sibun, P. and Reynar, J. C. (1996). Language determination: Examining the issues. In Proceedings of the 5th Annual Symposium on Document Analysis and Information Retrieval, pages 125–135.

Sun, X., Wang, H., Xiao, Y., and Wang, Z. (2016). Syntactic parsing of web queries. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1787–1796. Association for Computational Linguistics.

Travadi, R., Van Segbroeck, M., and Narayanan, S. (2014). Modified prior i-vector estimation for language identification of short duration utterances. In Proceedings of Interspeech.

Vatanen, T., Vayrynen, J. J., and Virpioja, S. (2010). Language identification of short text segments with n-gram models. In Proceedings of LREC 2010, pages 423–3430.

Wang, Z., Wang, H., and Hu, Z. (2014). Head, modifier, and constraint detection in short texts. In Proceedings of the International Conference on Data Engineering, pages 280–291, 03.

Zhang, Y., Riesa, J., Gillick, D., Bakalov, A., Baldridge, J., and Weiss, D. (2018). A fast, compact, accurate model for language identification of codemixed text. arXiv:1810.04142v1.