Do You Speak London?
Language Identification Platform

Tarek Amr (@gr33ndata)

1 Introduction

Do You Speak London(dysl) is a command line tool and python library for natural language identification, also known as LangID. Currently pre-packaged with training data for 4 languages; English, French, Spanish and Portuguese. However, you can simply re-train it on your own dataset.

2 Software Architecture

The core functionality of the code used in *dysl* is inherited from *IRLib (Information Retrieval Library)*. IRLib is written in Python and is available as Free and Open Source Software on-line (https://github.com/gr33ndata/irlib).

The implementation of the n-gram Language Model is explained here, while the next section is dedicated to the theoretical background for the n-gram Language Models.

The LM populated (trained) using add_doc(), which takes the following parameters:

- doc_id
- doc_terms

The doc_id here stands for the class-label, or language of document being read in our case here. A separate LM is created for each unique doc_id. Therefore, if more than one document are given the same doc_id, they all are used to train the same LM. doc_terms is a list of all characters in a document being read. In other words, documents are split into characters and the resulting list of characters is passed as doc_terms. In case of word-based LM, a list of words should be given to doc_terms rather than characters.

When LM is being instantiated, the following parameters can be set:

- n: The value of n in n-gram LM. Default is 3.
- lpad and rpad: Left and right padding. When empty string is given, then no padding is carried, otherwise, document are padded with (n-1) characters of the character values given to lpad and rpad. For example, in a 5-gram character-based LM, if lpad='A' and rpad='Z'. Then the string 'hello' will become 'AAAAhelloZZZZ' after padding. Default is empty string.
- smoothing: Default is 'Laplace' for add one smoothing as explained earlier. Additional discounting techniques are under development.
- laplace_gama: Sets the value of γ as explained in equation 5. Default is 1.
- corpus_mix: If given any value between 0 and 1, then it is used to set the value of λ as explained in equation 6. Default is 0, i.e. only term probabilities from documents are used. Additionally, if set tp 'l', then a log-likelihood odds (LLO) model is used.

For *dysl*, we set n=3, smoothing='Laplace', laplace_gama=0.1 and corpus_mix='l'.

2.1 classification

During the classification, the calculate() method of the Language Model is used. It only takes one parameter, doc_terms, which accepts data in the same format for the doc_terms parameter given to add_doc(). The calculate() returns a dictionary, with 'calc_id' key, which holds the value of the language being detected.

2.2 Social Language Model

To make it easier to identify the language of text being sent on social media, we are having another class SocialLM which uses LM as base class. For the training, add_doc() is kept unchanged, while SocialLM offers classify() as an alternative to calculate(). The resulting value for classify() is a tuple containing the id of the language being detected, and another parameter called 'karbasa', which is a Arabic word for crowded. Basically, the low 'karbasa' score, the more close the score of the identified language is to the scored of the closest language. This can be used as indicator to tell if our model is trying to classify a language it has never seen before during the training phases. Empirically, a 'karbasa' of lower value that 0.05 may mean that our model is confused.

Two more important methods in SocialLM are tokenize() and normalize(). The tokenize() method converts text into a string of characters (or words), while normalize() is capable of removing links and social media @mentions from text, and also applying UTF-8 NFC normalization.

3 Technical Details

We apply a similar approach to that used by Peng et al. (2003). We use here a character-based *n-Gram Language Model (LM).*

3.1 n-Gram Language Model

Let us assume we have a set of documents $D = \{d_1, d_2, ..., d_m\}$, and a set of classes $C = \{c_1, c_2, ..., c_k\}$, where each document is classified as member of one of these classes. For any document, d_i, the probability that it belongs to class c_j, can be represented as $Pr(c_j/d_i)$ and using Bayes rules (Zhai and Lafferty, 2001; Peng et al., 2003), this probability is calculated as follows:

$$Pr(c_j/d_i) = \frac{Pr(d_i/c_j) * Pr(c_j)}{Pr(d_i)} \tag{1}$$

The term $Pr(d_i)$ is constant for all classes. The term $Pr(c_j)$ can either represent the relative frequency of class j in the training set, or a uniform class distribution can be assumed, thus, we end up with the term, $Pr(d_i/c_j)$ (Grau et al., 2004). For a document d_i, that is composed of a sequence of words $w_1, w_2, ..., w_L$, $Pr(d_i/c_j)$ is expressed as follows; $Pr(w_1, w_2, ...w_L/c_j)$. We are going to write it as $Pr_{c_j}(w_1, w_2, ...w_L)$ for simplicity.

$Pr_{c_j}(w_1, w_2, ...w_L)$ is the likelihood that $w_1, w_2, ..., w_L$ occurs in c_j. This can be calculated as shown in equation 2.

$$Pr_{c_j}(w_1, w_2, ...w_L) = \Pi_{i=1}^{L} Pr_{c_j}(w_i/w_{i-1}, w_{i-2}, ..., w_1) \tag{2}$$

Nevertheless, in practice, the above dependency is relaxed and it is assumed that each word w_i is only dependant on the previous $n - 1$ words (Peng et al., 2003). Hence, equation 2 is transformed to the following equation:

$$Pr_{c_j}(w_1, w_2, ...w_L) = \Pi_{i=1}^{L} Pr_{c_j}(w_i/w_{i-1}, w_{i-2}, ..., w_{i-n+1}) \tag{3}$$

The n-gram model is the probability distribution of sequences of length n, given the training data (Manning and Schütze, 1999). Therefore, $Pr_{c_j}(w_1, w_2, ...w_L)$ is referred to as the n-gram language model approximation for class c_j. Now, from the training set and for each class, the n-gram probabilities are calculated using the maximum likelihood estimation (MLE) shown in equation 4 (Chen and Goodman, 1996):

$$\begin{aligned} Pr_{c_j}(wi/w_{i-n+1}^{i-1}) &= \frac{Pr(w_{i-n+1}^{i})}{Pr(w_{i-n+1}^{i-1})} \\ &= \frac{count(w_{i-n+1}^{i})/N_w}{count(w_{i-n+1}^{i-1})/N_w} \\ &= \frac{count(w_{i-n+1}^{i})}{count(w_{i-n+1}^{i-1})} \end{aligned} \tag{4}$$

Where N_w is the total number of words.

We are proposing to use the n-Gram Language model for URL-based classification. However, in our case here, we use characters instead of words as a basis of the language model. We construct a separate LM for each class of URLs as follows. The above probabilities are calculated for each class in the training set by counting the number of times all sub-strings of lengths n and $n - 1$ occur in the member URLs of that class. For example, we have the following strings as members of class c_j, {'ABCDE','ABC','CDE' }. In a 3-gram LM, we will store all sub-strings of length 3 and those of length 2, along with their counts, as shown in table 1.

Table 1: Sample data-structure for a 3-gram LM counts

3-grams	('ABC': 2), ('BCD': 1), ('CDE': 2)
2-grams	('AB': 2), ('BC': 2), ('CD': 2), ('DE': 2)

Counts in table 1 are acquired during the training phase. Then in the testing phase, URLs are converted into n-gram, and for each n-gram, its probability is calculated using equation 4. For a new URL, URL_i, it is classified as member of class c_j, if the language model of c_j maximizes equation 1, i.e. maximizes $Pr(c_j/URL_i)$.

3.1.1 Language Model Smoothing

The maximum likelihood in equation 4 can be zero for n-grams not seen in the training set. Therefore, smoothing is used to deal with the problem by assigning non-zero counts to unseen n-grams. Laplace smoothing is one of the simplest approaches (Jeffreys, 1998). Chen and Goodman (1996) explained that it is calculated as shown in equation 5.

$$Pr_{c_j}(wi/w_{i-n+1}^{i-1}) = \frac{count(w_{i-n+1}^{i}) + 1}{count(w_{i-n+1}^{i-1}) + V} \tag{5}$$

The count is increased by 1 in the numerator, and by V in the denominator, where V represents the number of unique sequences of length $n - 1$ found in the training set. Since the overall probabilities sum to 1, by using such smoothing, we are effectively taking a portion of the probability mass assigned the encountered sequences and reassigning the discounted value to the unseen sequences (Jurafsky and Martin, 2000). Laplace smoothing is also known as *add one* smoothing, since 1 is added to $count(w_{i-n+1}^{i-1})$. Rather than adding 1, both 1 and V can be multiplied by a coefficient γ in order to control the amount of the probability mass to be re-assigned to the unseen sequences.

Gale and Church (1994) highlighted that the equation 5 can be seen as follows: The value of $count(w_{i-n+1}^{i})$ in equation 4 is being replaced by $(1 + count(w_{i-n+1}^{i})) * \frac{N}{N+V}$, where N is equal to $count(w_{i-n+1}^{i-1})$ here. Similarly, for unseen sequences, the value will be estimated as follows: $\frac{N}{N+V}$. Then, they added that the assumption of having such geometric sequence doesn't usually hold for real data. Thus, more sophisticated smoothing techniques, such as Good-Turing discounting (Good, 1953), is sometimes preferred for estimating the probabilities of unseen sequences.

One problem with the smoothing techniques discussed earlier is that they give equal probabilities to all unseen n-grams. Chen and Goodman (1996) explained that the two bi-grams "burnish the" and "burnish thou" will be given the same probabilities provided the fact that neither of them appear in the training data. Whereas, the former seems to be more likely since 'the' is a more common word that 'thou'. In order to overcome this, Ponte and Croft (1998) suggested using the whole collection in order to estimate the likelihood of missing terms in documents - the missing n-grams in classes in our case. For a term t, its collection MLE can be calculated as follows: $Pr_{collection}(t) = cf_t/cs$, where cf_t is the raw count of t in the collection and cs is the total number of tokens in the collection. Song and Croft (1999) listed two formulae for combining terms probabilities in both documents and collection: the weighted sum (also known as linear interpolation, shown in equation 6) and the weighted product approach (shown in equation 7).

$$Pr_{sum}(t/c_j) = \lambda * Pr_{collection}(t) + (1 - \lambda) * Pr_{doc}(t/c_j) \tag{6}$$

$$Pr_{sum}(t/c_j) = Pr_{collection}(t)^{\lambda} * Pr_{doc}(t/c_j)^{(1-\lambda)} \tag{7}$$

where λ is a weighting parameter and it takes values between 0 and 1. $Pr_{doc}(t/c_j)$ is calculated the same way as in equation 4. As for the collection probabilities, $Pr_{collection}(t)$, there are 2 variations for calculating them, depending on which model is being used. In the model proposed by Miller et al. (1999), it is the raw count of t in the collection divided by the total number of tokens in that collection. The same as in the previous paragraph. Whereas in the model proposed by Hiemstra (1998), they treated query terms as a binary vector over the entire vocabulary (Lavrenko and Croft, 2001), hence, the collection probabilities were seen as the number of documents containing t divided by the total number of documents. This enabled them to justify how their model incorporates the classical term frequency and inverse document frequency weights of vector space model (Kraaij, 2004, p. 51). We use the model proposed by Miller et al. (1999) here.

Additionally, we use the weighted sum presented in equation 6 in the experiments presented here. However, Ponte and Croft (1998) warned that attention should be taken when mixing collection probabilities with those of the documents, as in some cases, common terms in a homogeneous collection might be given higher probabilities in documents they do not occur in compared to those they do occur in. Thus, Hiemstra (1998) elaborated that λ should not exceed 0.5, where Zhai and Lafferty (2001) added that smaller values of λ (≈ 0.1) are better for short documents.

In a similar fashion, there exists additional smoothing approaches, e.g. *Jelinek-Mercer* smoothing (Bahl et al., 1983), where the probabilities of n-grams of higher order of n are being interpolated with the probabilities of lower order n-grams.

3.1.2 Linked Dependence Assumption

In the n-gram LM, in order to move from equation 2 to equation 3, we need to assume that the probability of w_i depends only on that of the previous $n-1$ terms. Similarly, in the uni-gram LM, all terms are assumed to be totally independent, i.e. it is equivalent to a *bag of words*. Although, increasing the value of n relaxes the *independence assumption*, it is still a strong assumption to be made. Cooper (1995), pointed out the *linked dependence assumption* (LDA) as a weaker alternative assumption. Lavrenko (2009) explained the *linked dependence* as follows. Consider the case of a two words vocabulary, $V = \{a, b\}$. In the case of two classes, c_1 and c_2, and under the *independence assumption*, $Pr_{c1}(a, b) = Pr_{c1}(a) * Pr_{c1}(b)$. Similarly $Pr_{c2}(a, b)$ is the product of $Pr_{c2}(a)$ and $Pr_{c2}(b)$. Otherwise, when terms are assumed to be dependant, $Pr_{c1}(a, b)$ and $Pr_{c1}(a, b)$ can be expressed as follows:

$$Pr_{c_j}(a, b) = K_{c_j} * Pr_{c_j}(a) * Pr_{c_i}(b) \tag{8}$$

where K_{c_j} measures the dependence of the terms in class c_j. Terms are positively correlated if $K_{c_j} > 1$, and they are negatively correlated if $K_{c_j} < 1$. As mentioned earlier, with the independence assumption, K_{c_j} is equal to 1. Now, in Cooper's LDA, K_{c_j} is not assumed to be equal to 1, however it is assumed to be the same for all classes, i.e. $K_{c_1} = K_{c_2} = K_{c_j} = K$

Accordingly, the value of K might not be needed if we try to maximize the log-likelihood ratio of relevance of $Pr(c_j/d_i)/Pr(\bar{c}_j/d_i)$, rather than $Pr(c_j/d_i)$ as in equation 1, where $Pr(\bar{c}_j/d_i)$ is the posterior probability of all the other classes except c_j.

Hence, the equation of the new log-likelihood odds classifier is as follows:

$$\begin{aligned} logLL_{c_j} &= log(\frac{Pr(c_j/d_i)}{Pr(\bar{c}_j/d_i)}) = log(\frac{Pr(d_i/c_j) * Pr(c_j)}{Pr(d_i/\bar{c}_j) * Pr(\bar{c}_j}) \\ &= \Sigma_{i=1}^{L} log(\frac{Pr_{c_j}(w_{i-n+1}^{i})}{Pr_{\bar{c}_j}(w_{i-n+1}^{i})}) + log(\frac{Pr(c_j)}{Pr(\bar{c}_j)}) \end{aligned} \tag{9}$$

For a new URL, URL_i, it is classified as member of class c_j, if the language model of c_j maximizes equation 9, i.e. maximizes the $logLL_{c_j}$.

Additionally, the motivation for using the log-likelihood odds comes form (Lafferty and Zhai, 2003). In the field of *information retrieval*, a user enters a query Q, and each document D is to be ranked and retrieved according to that ranking. In the LM approach, such as the one proposed by (Ponte and Croft, 1998), a document D is ranked base on the probability assigned to Q according to the model calculated from D. On the other hand, in the *binary independence model* (BIM) (Robertson and Jones, 1976; Sparck Jones et al., 2000), the documents are subdivided into relevant R and non-relevant $\bar{R}$ documents, and the retrieval problem is dealt with in a way similar to a binary classification problem. A document D is ranked according to the log-likelihood odds of it relevance, $log(Pr(R/D)/Pr(\bar{R}/D))$. Thus, one reproach that has been laid upon the LM is that it does not incorporate the notion of relevance (Nallapati, 2004). In the LM discussed in section **??**, d_i and c_j are equivalent to Q and D respectively, however, since it is a classification problem that is being dealt with here, rather an information retrieval problem, for a document to be a member of c_j, this is also equivalent to it being relevant. Therefore, the log-likelihood odds in equation 9, is equivalent log-likelihood odds of relevance presented by

Lafferty and Zhai (2003), which shows that this new variation of LM can implicitly incorporate the notion of relevance. Additionally, the log-likelihood odds is commonly in classification tasks, e.g. Terra (2005); **?**.

Hereafter, we refer to this variation of the n-gram LM as *Log-likelihood Odds* (LLO) model. It is worth mentioning that the use of logarithmic scale also helps in preventing decimal point overflow during the implementation.

References

Bahl, L. R., Jelinek, F., and Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):179–190.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

Cooper, W. S. (1995). Some inconsistencies and misidentified modeling assumptions in probabilistic information retrieval. *ACM Transactions on Information Systems (TOIS)*, 13(1):100–111.

Gale, W. and Church, K. (1994). What is wrong with adding one. *Corpus-based research into language*, pages 189–198.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264.

Grau, S., Sanchis, E., Castro, M. J., and Vilar, D. (2004). Dialogue act classification using a bayesian approach. In *9th Conference Speech and Computer*.

Hiemstra, D. (1998). A linguistically motivated probabilistic model of information retrieval. In *Research and advanced technology for digital libraries*, pages 569–584. Springer.

Jeffreys, H. (1998). *The theory of probability*. Oxford University Press.

Jurafsky, D. and Martin, J. (2000). *Speech & Language Processing*. Pearson Education India.

Kraaij, W. (2004). *Variations on language modeling for information retrieval*. PhD thesis, University of Twente, Enschede.

Lafferty, J. and Zhai, C. (2003). Probabilistic relevance models based on document and query generation. In *Language modeling for information retrieval*, pages 1–10. Springer.

Lavrenko, V. (2009). *A generative theory of relevance*, volume 26. Springer.

Lavrenko, V. and Croft, W. B. (2001). Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM.

Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*, volume 999. MIT Press.

Miller, D. R., Leek, T., and Schwartz, R. M. (1999). A hidden markov model information retrieval system. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 214–221. ACM.

Nallapati, R. (2004). Discriminative models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71. ACM.

Peng, F., Huang, X., Schuurmans, D., and Wang, S. (2003). Text classification in asian languages without word segmentation. In *Proceedings of the sixth international workshop on Information retrieval with Asian languages-Volume 11*, pages 41–48. Association for Computational Linguistics.

Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM.

Robertson, S. E. and Jones, K. S. (1976). Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146.

Song, F. and Croft, W. B. (1999). A general language model for information retrieval. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321. ACM.

Sparck Jones, K., Walker, S., and Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments: Part 1. *Information Processing & Management*, 36(6):779–808.

Terra, E. (2005). Simple language models for spam detection. In *TREC*.

Zhai, C. and Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM.