As indicated earlier, we have divided the problem into three parts. These are monolingual identification, language enumeration and segment identification. We make the *limited ambiguity assumption* (only two languages per document), assume a high level of diversity, and also assume that the language can shift at every word, which is a very realistic assumption. The last assumption means that we have to identify the language of every word in the document.

The outline of our method is shown in Figure 2. We first train a monolingual identifier as described in the previous section. Then, given a test document, we split it into words and form lists of word types as well as word tokens (instances of word types). Then, through a number of iterations which could be two or more depending on the level of diversity, number of confusable pairs, etc., we enumerate or identify the languages present in the document.

Category	Count	Examples
Related	51	Assamese-UTF8::Oriya-UTF8 Danish-ISO-8859-1::Norwegian-ISO-8859-1 etc.
Less Related	41	Catalan-ISO-8859-1::Russian-Windows-1251 Punjabi-UTF8::Telugu-UTF8 etc.
Unrelated	91	Dutch-ISO-8859-1::Marathi::Saamanaa Hindi-Typewriter::Tagalog-ISO-8859-1 etc.

Table 2. Language-Encodings Tested on for Evaluation

6.1. Language Enumeration

Two algorithms are being used at this stage. One is *monoKBest* (Algorithm 2). The other is *languageEnumerator* (Algorithm 1). The *monoKBest* algorithm returns K-best n-gram models or language-encodings for a given word type or token. The *languageEnumerator* algorithm returns the m-best language-encodings for a given document.

6.1.1. Notations Used in Algorithms

The following are the notations used in the algorithms:

1. d is a multilingual document.
2. $L = (l_1, \ldots, l_j, \ldots, l_J)$ is the list of possible language-encodings for any of the words in a given document. Note that J represents the global diversity assumption.
3. m is the number of languages after enumeration. It represents the local diversity assumption.
4. K is the number of best possible language-encodings in each iteration.
5. $MLS(x_i)$ is a vector of language-encodings and their corresponding scores for the word x_i:
   $MLS(x_i) = ((l_1^i, s_1^i), (l_2^i, s_2^i), \ldots, (l_K^i, s_K^i))$

6. $\overrightarrow{W} = (w_1, w_2, \ldots, w_r, \ldots, w_K)$ is a weight vector, where w_r is the weight assigned to a language-encoding (l), if l is the r^{th} best language-encoding of the word x_i. The values in this vector are manually assigned constants decreasing exponentially with the rank (p). These weights can be used to tune the algorithm.

Algorithm 1 languageEnumerator(d, K, L, W, m)

1: **for** each word $x_i \in d$ **do**
2: $MLS(x_i) \leftarrow monoKBest(x_i, L, K)$
3: **end for**
4: Total score of l_j in a document d is
$S_j \leftarrow \sum_{x_i} W_k * S_k^i$, where $l_j \in L$ and (l_k^i, S_k^i) occurs in the vector $MLS(x_i)$ for some $x_i \in d$ and $l_k^i = l_j$
5: $returnList \leftarrow < l_j, S_j >$
where *retunList* contains l_j, S_j pairs
6: Sort *returnList* based on the scores
7: $L \leftarrow K$-best languages-encodings (l_j's) from the *returnList*
8: **if** $K == m$ **then**
9: return *returnList*
10: **else**
11: $languageEnumerator(d, K-1, L, W, m)$
12: **end if**

Algorithm 2 monoKBest(x_i, L, K)

1: $MLS(x_i) \leftarrow K$-best possible n-gram models (language-encodings) l_k^i and their corresponding scores for the word x_i, where $l_k^i \in L$.
2: Sort MLS based on n-gram model scores.
3: return $MLS(x_i)$.

In each iteration for language enumeration, we go through the following steps:

1. Using the monolingual identifier, select the K-best n-gram models, *i.e.*, language-encoding classes for each word type or token.
2. Assign some weight to each of these classes depending on the relative rank of the class, *i.e.* assigning the weights to weight vector $\overrightarrow{W}$. What is crucial here is to assign the best weights to classes according to their ranks. To see why this is important, consider a document that has mostly Hindi words with a few English words. English and Hindi are the correct language classes present in this document. When the language classes are computed for Hindi words, the next best estimate of classes for these words is Marathi because it is very similar to Hindi. Thus, a large number of words in this document have Marathi as their second rank language. English appears as a first rank language for very few words. If we choose linearly decreasing weights for ranks, Marathi may overtake English in the cumulative score due to the sheer number of words falling into this class. However, if we