Language Identifier: A Computer Program for Automatic Natural-Language Identification of On-line Text

Kenneth R. Beesley
Address from 1988:
Automated Language Processing Systems
(a.l.p. Systems)
190 West 800 North
Provo, UT 84604
USA

Originally published in
Languages at Crossroads:
Proceedings of the 29th Annual Conference
of the American Translators Association,
12-16 October 1988, pp. 47-54.

Address 1999:
Xerox Research Centre Europe
6, chemin de Maupertuis
38240 MEYLAN, France
ken.beesley@xrce.xerox.com

Abstract

Keywords: Language Identification, Machine Translation, Computer Translation, Machine-Assisted Translation, Computer-Assisted Translation, Cryptology, Cryptanalysis.

The first step in translating any text is to identify the language in which it is written. Several useful methods have already appeared for language identification where the mystery texts are properly spelled and accented paper documents. Unfortunately, in machine-translation environments, where texts are on-line and may exhibit a variety of conventions for character-mapping and accentuation, the problem is far more difficult. This paper outlines a generalized approach to language identification of on-line text based on techniques of cryptanalysis. A working prototype has been built, and the results are promising.

1 The Language Identification Problem

For most translation jobs, the first step of language identification is resolved so quickly that it is virtually ignored. Given, however, that there are an estimated three or four thousand languages in the world, even professional translators are sometimes stumped by mystery texts. Recently, a.l.p. Systems found that this can be an especially challenging problem for machine-translation systems, where texts are on-line, and we wrote a program called the Language Identifier to perform automatic language identification.

Preliminary research has found a few papers by translators on language identification. There appear to be two basic strategies. One provides lists of letters, especially accented letters, that appear in various languages (Keesan, Ref. 1; Newman, Ref. 2; Bokor, Ref. 3). By matching up the letters found in a document with the letters in an identification list, the language can often be identified quite quickly and easily. The second strategy provides lists of short words that appear in various languages—one then matches short words in the document with short words in the list to find the language (Ingle, Ref. 4). Individual methods may include mixtures of these strategies and also some examples of typical word endings for various languages. These methods, which I will call "translator approaches," are usually formalized in short tables.

The translator approaches are interesting and useful in their intended applications, but they are not directly applicable to on-line text. The problem lies in the following assumptions:

A. Translator approaches require a human evaluator manually to compare a document with the information in the tables. The purpose of the Language Identifier is to automate this process for on-line text. In addition, the translator approaches require the mystery document to be available in standard printed form; that is, the text must be completely and correctly displayed. On-line text, however, is stored in computer memory, where each letter is actually stored as a fairly arbitrary integer value. In general, the computer and its software will be able to display or print the integer values correctly only AFTER the language has been identified.

B. Translator approaches assume that the mystery text is correctly spelled, including any accentuation. In fact, the presence or absence of certain kinds of accentuation is often the vital key in the tables. On-line text, on the other hand, may have separate integers to represent accented letters, may represent accented letters with arbitrary integer sequences, or may have the accentuation completely deleted or neutralized. The accented letter 'á', for example, may simply be represented with the same integer as for unaccented 'a', however incorrect this may be from a formal point of view. Deaccented French, Spanish,

and Portuguese are not at all uncommon inside computer systems. The lack of accents can result from laziness during data entry, the lack of accenting capabilities in the hardware or software, or from entering the text in all capital letters.

C. In addition, the automatic Language Identifier will have to handle texts that have been entered into the computer with a variety of character-to-integer mappings. EBCDIC and ASCII represent two such mappings. For example, the ASCII representation of the letter 'A' is the decimal integer 65 while the EBCDIC representation is decimal 193. In the end, such mappings are really quite arbitrary, and compatibility among computers depends on different manufacturers adhering to a common standard or providing conversion programs to map from one standard to another.

   There are also different "flavors" of EBCDIC and ASCII for representing languages other than English. The decimal value 123, for instance, is used to represent '#' in the English flavor of EBCDIC, 'Ñ' in Spanish, 'Ã' in Continental Portuguese, 'Õ' in Brazilian Portuguese, and 'Æ' in Danish/Norwegian, to mention just a few possibilities.

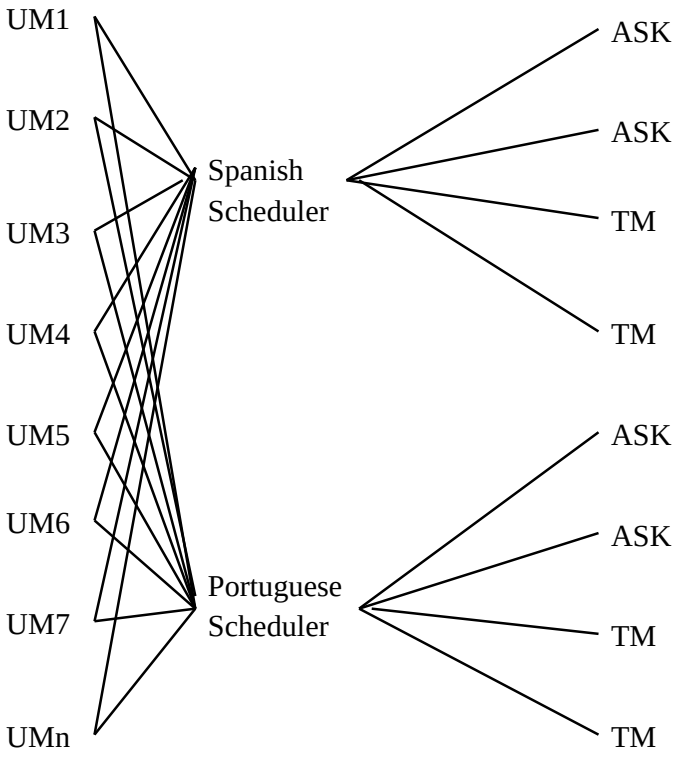
   Even worse, the internal mappings for exotic languages, such as Arabic, can differ completely with each machine or word processor program. To handle such diversity, a scheme far more open-ended and flexible than the translator tables will be needed. It goes without saying that the program cannot be told in advance which mapping convention was used—the whole idea of the Language Identifier is to identify the language of genuine mystery texts.

D. The short-word approach assumes that you have a complete running text with short words to identify. In practice, automatic language identification will have to deal with sample texts that lack short words. This could happen when the sample text is very short, perhaps a single word, or when the sample is telegraphic or just a "word salad."

E. Translator approaches collect ad-hoc features to look for; they provide no mechanism for adding new languages to the lists. The Language Identifier must be generalized to allow reasonably competent users to add languages with a minimum of effort.

2 Language Identification at a.l.p. Systems

A.l.p. Systems is an established vendor of computer-assisted translation software and is the largest translation-services company in the world. We are perhaps best known for our interactive translation programs AutoTerm and TransActive, which are packaged in translation workstations and which are



available for a number of language pairs. Because such interactive translation workstations are used by skilled translators, usually in production environments, language identification has never been much of a problem.

More recently, a.l.p. Systems has started building totally automatic translation programs that can be interfaced to almost any user environment. One such program, called ASK, receives a source text and performs baseform reduction and dictionary look-up, returning lists of possible translations for each term. A more ambitious program, called TransMatic, goes on to perform syntactic analysis, transfer and synthesis for each sentence, returning first-draft translations. So far, ASK and TransMatic have been built for Spanish-to-English and Portuguese-to-English, and other language pairs are in planning.

An ASK/TransMatic configuration looks like this [see Figure 1], where a UM is a user machine and a TM is a TransMatic machine. These "machines" are not separate pieces of hardware but virtual machines in an IBM mainframe computer. There might be dozens or hundreds of user machines, which could be editors, scanners, data-base drivers, or almost anything. There may also be numerous scheduler machines, but each one is dedicated to a specific language pair. Each scheduler may, in turn, control up to 50 slave machines, each containing ASK and/or TransMatic for the particular language pair.

Technically speaking, each scheduler and slave is a disconnected virtual machine, and all the machines communicate back and forth using VMCF (Virtual Machine Communication Facility), a standard IBM interface protocol for virtual machines.

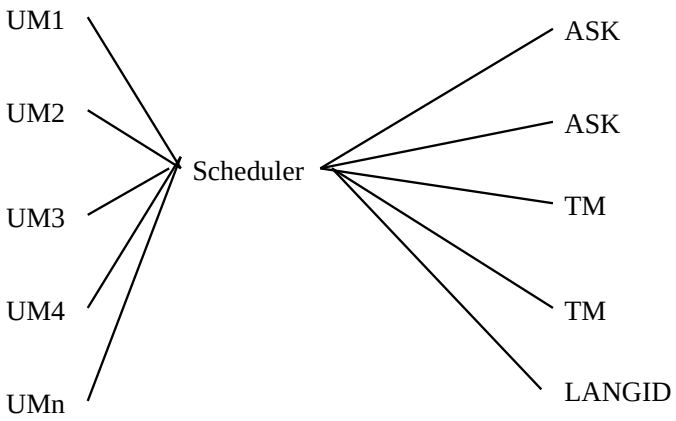
In a typical scenario, a user might be scanning a file using IBM XEDIT and find a sentence that he or she wants translated. The user then invokes user-supported software that generates a VMCF translation request and directs it to an appropriate scheduler. The request will include a copy of the text to be translated, anything from a single word to an entire file, and a number of parameters set as desired by the user. The scheduler knows which, if any, slaves are available, and it either queues the job or immediately forwards it to a slave for translation. The ASK or TransMatic slave performs the translation and then communicates the results back to the user.

AutoTerm and TransActive, the traditional interactive translation programs written by a.l.p. Systems, are designed to assist a skilled translator to produce polished translations. ASK and TransMatic are designed with very different assumptions: First, ASK and TransMatic users do not have to be translators at all—they might not know a single word of Spanish or Portuguese. Second, ASK and TransMatic work totally automatically rather than interactively. Third, ASK and TransMatic output is intended for information gathering, also called "gisting," or "scanning"; that is, they are primarily designed to give the user a good idea of what the text is about, not to produce or assist in producing polished output.

In practice, we found that many users and potential users of ASK and TransMatic were indeed ignorant of Spanish and Portuguese; in addition, and somewhat to our surprise, we soon discovered that such users were also unable to distinguish between Spanish and Portuguese. They innocently sent Spanish text to the Portuguese scheduler and vice versa, or they were consciously frustrated when faced with the choice. We had overlooked the first problem of translation, language identification. Automatic, on-line language identification suddenly became desirable.

3 Design and Prototype

In October and November of 1987, I designed and built a prototype of the Language Identifier. In simple overview, it functions much like an ASK or TransMatic program. It is passed a buffer of mystery text, of arbitrary size, from the user environment, and it returns a buffer of information indicating the natural language in which the text is written. The prototype is written in the C language and presently runs on Data General hardware.



3.1 Configuration

In a production environment, a Language Identifier, or several of them, can be enslaved to one or more schedulers, just like ASK and TransMatic. Users who want a text translated, but who don't know the source language, can then send the text off first for language identification rather than translation.

3.2 Theoretical Foundations

The Language Identifier is based on mathematical source-language models that have been developed in the field of cryptanalysis for help in breaking ciphers (Konheim, Ref. 5). The prototype has shown that these mathematical models can be successfully modified, reinterpreted and reapplied to the problem of natural-language identification.

Automatic language identification is possible because alphabetically written natural languages are highly non-random and consistent in the letters and letter sequences that they use; and equally important, different languages differ consistently in the letters and letter sequences used. In other words, each language uses a unique or very characteristic alphabet, and letters in the alphabet appear with surprisingly consistent frequencies in any statistically significant text; in addition, the frequencies of occurrence of sequences of two, three, four, five and more letters are characteristically stable within, and diverse among, natural languages. (These insights are hardly original: Kahn, in his excellent history of cryptology, points out that they appeared, in essence, in an Arabic encyclopedia written by Qalqashandi in 1412, and Qalqashandi attributes most of his information to Ibn ad-Duraihim, who lived from 1312 to 1361. See Ref.6.)

For example, standard English text contains the following letter characters in both upper-case and lower-case.

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Standard Spanish is characterized by the presence of the following letters:

ñ á é í ó ú ü

(Spanish digraphs like 'll' and 'ch' are counted as single letters for purposes of alphabetization but not, usually, for representation, either inside or outside the computer.)

Standard Portuguese is characterized by the following:

á é í ó ú
ã õ
â ê ô
à
ü
ç

Where text is correctly spelled, even inside a computer, the presence or absence of characteristic letters (or the integer mappings) is often quite sufficient for language identification.

When accentuation is lacking, the relative frequencies of individual letters are often revealing. In English, for example, the letter 'E' accounts for about 13% of the average text; 'U' accounts for about 3% and 'Z' for about 0.1%. These figures, usually recast in terms of probabilities of occurrence, are fairly stable for English and quite unique to English. 'U' and 'Z', for example, occur more frequently in German than in English. Going one step further, one can compute the probability of occurrence of two-letter sequences or "digrams." For example, the probability of "TH" occurring in English is relatively high, but in Spanish or Portuguese the probability approaches zero; the probability of "SZ" is relatively high in Polish and Hungarian but low in English, French, Spanish, and Portuguese. Such information can be quantified precisely for a representative corpus of the language, and one can even go on to compute probabilities for 3-grams, 4-grams, etc. In doing, so, the traditional observations about typical letters patterns fall out: e.g. that the sequence "tion" is typical of English and French, while "ción" is typical of Spanish and "ção" of Portuguese, etc. The Language Identifier literally computes and stores thousands of items of information for each language, providing a statistical model of how likely each letter and letter sequence is to occur in each language.

Using such information about letter and sequence probabilities for each language, the Language Identifier takes each word in the mystery text and computes the probability that it is English, the probability that it is French, the probability that it is Spanish, etc., for all the languages in the library. The language with the highest probability wins. In effect, the Language Identifier allows each language model to compete to see which one is most likely to account for the whole mystery sample.

3.3 Generality and Expandability

For such a system to be truly useful, the addition of a new language to the program's library should require no new coding, no special training, no formal-linguistic expertise, and no dictionary building. What the Language Identifier does require for each new language is a consistent, representative corpus of the new language; but someone with reasonable facility in the new language should be able to build such a corpus in less than a week, perhaps within hours in some cases. Preliminary tests have shown that a 64K character corpus will be more than sufficient, and useful but inconsistent results have been achieved with corpora as small as 6K characters. The Language Identifier includes an add-language program that analyzes the corpus and automatically adds the necessary language-model information to the library.

Once suitable corpora have been built and analyzed, the Language Identifier is able to identify texts that are properly accented, deaccented, upper-case, lower-case, etc. The key to this ability is, somewhat ironically, to abondon any attempt to identify English, French, German, or whatever in any standardized sense. What the Language Identifier does is compare the orthographical features of a mystery text against the orthographical features of the languages in the library. These features are defined for each language by its corpus. There might, for example, be a half dozen or more "Spanish" corpora, each representing a different set of orthographical conventions and/or character-set mappings used to represent what we abstractly call Spanish. For the purposes of the Language Identifier, these would be separate languages, each with a separate, ideally descriptive, name chosen by the user. We might find the following:

A. a corpus of Spanish text all properly accented and lower-case: named **spanaclc**

B. a corpus of Spanish text all properly accented and upper-case: named **spanacuc**

C. a corpus of Spanish text, upper-case, where accented letters are represented as the sequences "/A," "/E," "/I," "/O," "/U," "*U" and the 'ñ' is represented as the sequence "~N": named **spanpreacuc**

D. As C, but lower-case: named **spanpreaclc**

E. a corpus of Spanish text, deaccented and lower-case: named **spandeaclc**

F. as E, but upper-case: named **spandeacuc**

Starting with one properly accented and capitalized corpus, all the others could be generated with simple conversion programs as the need arose. For

example, if Spanish-speaking users suddenly found that they were getting Spanish text with a new orthographical convention, such as representing the 'ñ' as the sequence "N#," it would be a trivial matter to generate a new representative corpus from spanpreacuc (letter C above) and add it to the library. Thereafter any text illustrating that orthographical convention would be identified as this particular variety of Spanish.

The identification of a text as spanpreaclc, or whatever, will actually be more helpful in most cases than a simple identification as "Spanish"; it can guide the user in setting the parameters for actual translation using programs like a.l.p. Systems ASK and TransMatic. The identification of a particular set of orthographical conventions may also help in pinpointing the provenance of a text.

4 Limitations

The Language Identifier is not infallible, all-purpose, or otherwise omnipotent. Expectations should be conditioned by the following caveats:

A. The Language Identifier at any given time "knows about" only those languages that are in the library.

B. The closer any two languages are, the harder it will be to distinguish between them. Closeness may result from common historical roots, borrowing, or overlapping character-to-integer mappings inside the computer. So far, however, even Spanish and Portuguese have been easy to distinguish.

C. The Language Identifier is designed to work for languages wherein the size of the alphabet is less than or equal to 256. That is, the language Identifier assumes that words are represented as strings of characters or, internally, integers, with one integer to a byte. This includes languages with spelling conventions that use digraphs or other sequences to represent accentuated characters or other exotic characters. Thus if the German 'ß' is represnted inside the computer as the sequence "~B," or if the Spanish 'ó' is represented as "/o," these orthographies can be handled well by the Language Identifier. Excluded, at least for the time being, are languages like Chinese where the "alphabet" numbers in the thousands and each letter (word) must be represented inside the computer with two or more bytes.

   Consistent romanizations or other phonemic alphabetizations of ideographic languages, such as Chinese Pinyin, can be handled by the Language Identifier. In general, romanizations of languages that usually use non-roman alphabets, such as Russian and Arabic, will also

be handled—such a change of alphabet will amount only to a new character-to-integer mapping inside the computer.

D. The Language Identifier works with probabilistic models, and so the results are always more or less probable; there will never be any absolute identifications, only more and less confident ones.

E. The reliability of any language model will be dependent on the size and internal consistency of the corpus from the model is derived.

F. For all the reasons described above, the correct identification of any single word is far less than certain. As the size of the mystery sample increases, however, the accumulation of identifications tends to point with increasingly certainly to a single language. For example, in an average sentence-sized sample we might find that 60 or even 70 percent of the words are identified as English, making English the "winner" overall. Preliminary indications are that ten to twelve words are sufficient for fairly confident identifications.

5 Conclusion

Using mathematical models from cryptanalysis, natural-language identification of on-line text is possible. Such a capability can certainly be useful in machine-translation systems, and it might also be used for various types of data analysis. A.l.p. Systems maintains both a commercial and a scientific interest in further developing this technology.

6 References

1. Keesan, Cynthia. "Identification of Written Slavic Languages" in *Proceedings of the 28th Annual Conference of the American Translators Association* (ed. Karl Kummer), 8-11 October 1987, pp. 517-528.

2. Newman, Patricia. "Foreign Language Identification—First Step in the Translation Process" in *Proceedings of the 28th Annual Conference of the American Translators Association* (ed. Karl Kummer), 8-11 October 1987, pp. 509-516.

3. Bokor, Gabor. "Romance Languages" in *Proceedings of the 1981 Conference of the American Translators Association* (reprint).

4. Ingle, Norman C. "Language Identification Table" (Shoreham, England 1986) privately printed and distributed. Also earlier version "A Language Identification Table" in **The Incorporated Linguist**, 15(4): 98-101, 1976.

5. Konheim, Alan G. *Cryptography: a Primer.* John Wiley & Sons, 1981.

6. Kahn, David. *The Codebreakers.* Macmillan, 1967.

Slide 1—Language Identifier—A.l.p. Systems Sample Analysis

Digram-Based Source-Language Model from Konheim

To add a language to the library:

1. Assemble a corpus in computer memory
2. Compute the total number of two-letter sequences (for demonstration purposes, letters will be used rather than numbers)
3. Compute the total number of occurrences of each distinct two-letter sequence or "digram" (e.g. AA, AB, AD,..., ZZ, etc.)
4. For each distinct two-letter sequence, divide the number of its occurrences by the total number of digrams in the corpus, yielding the Probability of Occurrence

Slide 2—Language Identifier—A.l.p. Systems

Computing Probabilities of Occurrence

One sample corpus of French contained 6085 total digrams and 288 distinct digrams. The digram "EP" appeared seven times; therefore the probability of occurrence of "EP" in the French corpus is 7 divided by 6085 or 0.001150.

$P_{Frn}(EP) = 7 \div 6085 = 0.001150$

Or, in equivalent mathematical terms, the digram "EP" accounts for 0.1150% of all the digrams in the corpus.

Slide 3—Language Identifier—A.l.p. Systems

Computing Probabilities of Occurrence

Similarly, the digram "EU" appeared 28 times and the sequence "SPACE-E" appeared 97 times, and so on for all the distinct digrams found in the corpus.

$P_{Frn}(EU) = 28 \div 6085 = 0.004601$

$P_{Frn}(\ E) = 97 \div 6085 = 0.015940$

"SPACE-E", occurring 97 times, is a relatively common digram: its probability of occurrence is fairly high. At the other end of the scale, "CB" never occurs at all, yielding a probability of occurrence of zero. And "CY" appears only once in the corpus, yielding a relatively low probability of occurrence that approaches zero.

$P_{Frn}(CB) = 0 \div 6085 = 0$

$P_{Frn}(CY) = 1 \div 6085 = 0.000164$

(Sample Digram Probability Data Slide)

```
83 85 0.001326 S U
83 89 0.001658 S Y
84 32 0.016417 T
84 65 0.003482 T A
84 69 0.009452 T E
84 70 0.000165 T F
84 72 0.025373 T H
84 73 0.008789 T I
84 76 0.000331 T L
84 79 0.006467 T O
84 82 0.003150 T R
84 83 0.002819 T S
84 84 0.000829 T T
84 85 0.002819 T U
84 87 0.000497 T W
84 89 0.000497 T Y
85 32 0.000165 U
85 65 0.002321 U A
85 67 0.002155 U C
85 68 0.000165 U D
85 69 0.000995 U E
85 71 0.000995 U G
85 73 0.000497 U I
85 76 0.001160 U L
85 77 0.001160 U M
85 78 0.003150 U N
85 79 0.000331 U O
85 80 0.000497 U P
```

Slide 4–Language Identifier—A.l.p. Systems

Probabilities of Occurrence

1. Probabilities of occurrence for each distinct digram are computed and stored FOR EACH LANGUAGE.

   $P_{Eng}(TH) = 0.025373$ $\quad$ $(relatively\ high)$

   $P_{Frn}(TH) = 0.000821$ $\quad$ $(rather\ low)$

   $P_{Esp}(TH) = 0$ $\quad$ $(did\ not\ appear)$

   $P_{Spn}(TH) = 0$ $\quad$ $(did\ not\ appear)$

2. The Language Identifier can combine such information to compute the probability that each word in a mystery sample is English, French, Esperanto, or whatever.

3. Scores for each word can be weighted together to yield the best guess for the mystery sample as a whole.

Slide 5—Language Identifier—A.l.p. Systems

Sample Language Identification

1. Assume we have a library of three languages, named English, French and Esperanto. All the digram probability information for these languages is computed and stored.

2. We have a mystery text inside the computer. It is stored as a string of integers, but for purposes of demonstration we will render it as the following:

   LES ORDINATEURS SONT APPELES A JOUER UN ROLE

3. Not knowing the language, we submit the text to the Language Identifier.

Slide 6—Language Identifier—A.l.p. Systems

The Language Identifier in Action

1. Divide each word into digrams. (In these examples, each word will be considered to start with a space, and, if necessary to complete the digrams, end with a space.)

2. Look up the probability of occurrence for each digram for each language in the library.

3. Multiply all the probabilities for a single language together, yielding the probability that the whole word belongs to that language.

Slide 7—Language Identifier—A.l.p. Systems

Computing Word Probabilities

```
Mystery text:

LES ORDINATEURS SONT APPELES A JOUER UN ROLE
```

The first word is LES—divide it up into two digrams

```
( L) (ES)
```

Look up the probabilities of occurrence:

$P_{Eng}(\ \ L) = 0.006799$
$P_{Eng}(ES) = 0.007794$

$P_{Frn}(\ \ L) = 0.019391$
$P_{Frn}(ES) = 0.026622$

$P_{Esp}(\ \ L) = 0.032301$
$P_{Esp}(ES) = 0.010654$

Multiply the probabilities together:

$P_{Eng}(LES) = 0.006799 \times 0.007794 = 0.000052991$

$P_{Frn}(LES) = 0.019391 \times 0.026622 = 0.000516227$

$P_{Esp}(LES) = 0.032301 \times 0.010654 = 0.000344134$

Slide 8—Language Identifier—A.l.p. Systems

Processing Continues

The second word is ORDINATEURS:

```
( O) (RD) (IN) (AT) (EU) (RS)
```

Look up the probabilities (here multiplied by 100 to avoid loss of precision)

$P_{Eng}(\ O) = 1.8905$
$P_{Eng}(RD) = 0.1492$
$P_{Eng}(IN) = 1.9568$
$P_{Eng}(AT) = 1.0945$
$P_{Eng}(EU) = 0$
$P_{Eng}(RS) = 0.1492$

$P_{Frn}(\ O) = 0.4437$
$P_{Frn}(RD) = 0.0657$
$P_{Frn}(IN) = 0.9202$
$P_{Frn}(AT) = 0.8052$
$P_{Frn}(EU) = 0.4601$
$P_{Frn}(RS) = 0.2136$

$P_{Esp}(\ O) = 0.1860$
$P_{Esp}(RD) = 0.1691$
$P_{Esp}(IN) = 0.7102$
$P_{Esp}(AT) = 0.2705$
$P_{Esp}(EU) = 0$
$P_{Esp}(RS) = 0.0338$

Total scores (multiply probabilities together):

$P_{Eng}(ORDINATEURS) = 0 \qquad (eliminated)$

$P_{Frn}(ORDINATEURS) = 0.0021227 \qquad (winner)$

$P_{Esp}(ORDINATEURS) = 0 \qquad (eliminated)$

Slide 9—Language Identifier—A.l.p. Systems

The analysis continues

$P_{Eng}(SONT) = 2.00457$ $(reasonable\ second)$
$P_{Frn}(SONT) = 4.05189$ $(winner)$
$P_{Esp}(SONT) = 0.05359$ $(poor\ third)$

$P_{Eng}(APPELES) = 0.022347$ $(reasonable\ second)$
$P_{Frn}(APPELES) = 0.054579$ $(winner)$
$P_{Esp}(APPELES) = 0$ $(eliminated)$

$P_{Eng}(A) = 0.017412$ $(winner)$
$P_{Frn}(A) = 0.009038$ $(reasonable\ third)$
$P_{Esp}(A) = 0.010147$ $(reasonable\ second)$

$P_{Eng}(JOUER) = 0.02458$ $(reasonable\ second)$
$P_{Frn}(JOUER) = 0.03277$ $(winner)$
$P_{Esp}(JOUER) = 0$ $(eliminated)$

$P_{Eng}(UN) = 0.31448$ $(reasonable\ second)$
$P_{Frn}(UN) = 0.68803$ $(winner)$
$P_{Esp}(UN) = 0.13633$ $(poor\ third)$

$P_{Eng}(ROLE) = 0.20159$ $(reasonable\ second)$
$P_{Frn}(ROLE) = 0.40071$ $(winner)$
$P_{Esp}(ROLE) = 0.09259$ $(poor\ third)$

Overall score for 8 words:

	Wins	Reasonable Losses	Eliminations
French	7	1	0
English	1	6	1
Esperanto	0	1	3