N-Gram Morphemes for Retrieval

Paul McNamee and James Mayfield
JHU Applied Physics Laboratory
{paul.mcnamee,james.mayfield}@jhuapl.edu

Abstract

Stemming, an approximation to morphological analysis, is a commonly used technique to improve performance in information retrieval systems. In the MorphoChallenge 2007 evaluation we applied a simple zero-knowledge technique that is based on frequency counts rather than machine learning. Our method is based on substituting a single fixed-length substring for each word that appears in documents or queries. We hope to discover whether this method, which has been used in previous IR evaluations with good effect, will be as effective for the information retrieval task as the unsupervised methods used by other participants. It should be emphasized that out submission was *not* a credible attempt to learn morphology and thus is not expected to perform well in the morphology induction task.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval

General Terms

Experimentation

Keywords

Morphological Analysis, Character N-gram Tokenization, Text Retrieval

1 Introduction

Our aim in the MorphoChallenge 2007 evaluation was to explore how a simple method for approximating word stemming would compare against state of the art morphology induction on the Information Retrieval task. Our method is based on the use of character n-gram tokenization, which we have used successfully in the Cross Language Evaluation Forum (CLEF) competitions. N-grams have the advantages of simplicity and accuracy, but do incur penalties of increased disk space requirements and longer run times. In an attempt to achieve the benefits of n-gram indexing without incurring a substantial cost in run-time performance we devised a method of 'n-gram stemming' that replaces each word with a selected n-gram. The method only presumes the availability of text, which is clearly reasonable for IR applications.

In Section 2 we explain n-gram indexing and describe how synthetic morphemes are obtained. We evaluate the effectiveness of the technique in Section 3 using previous CLEF data sets. We summarize our findings in Section 4 and give a complete listing of our source code for producing morphemes as an appendix.

2 Character N-gramming

2.1 N-gram Indexing

Commonly IR systems adopt a bag-of-words model where words in a document are treated as the basic indexing units. Often the tokenization process additionally transforms words into stems or root forms so that words that differ only by inflectional or perhaps even derivational morphology are treated as the same. With character n-gram indexing, words are not considered the basic indexing unit; instead, character substrings are used, typically of a fixed length. For example, for the text 'Johns Hopkins' the character 4-grams generated are: *#joh*, *john*, *ohns*, *hns#*, *ns#h*, *s#ho*, *#hop*, *hopk*, *opki*, *pkin*, *kins*, and *ins#* [1]. While short n-grams can have higher individual ambiguity (e.g., *ins#* could come from the words *fins*, *skins*, and *dolphins*, among others), this collection of twelve n-grams is not very likely to occur unless the phrase 'Johns Hopkins' was present in the original text. Storing 12 entries in an inverted file instead of just two (one for each word) is why n-gram indexing suffers from performance issues. However, this expense also enables flexible matching. Web log analysis shows that roughly 30% of users mistakenly[2] type "john hopkins" and yet 8 out of the 12 n-grams still match this string.

N-grams have been used in many Asian languages because of inherent difficulties with word segmentation; however they have not been very popular in Western languages. Damashek [1] promoted their use at early TREC evaluations, but his claims generated controversy and were received with skepticism. McNamee and Mayfield [3] overwhelmingly demonstrated that n-grams are effective in European languages and presented compelling evidence using eight languages from the CLEF 2002 evaluation. They found that lengths of n=4 or n=5 worked about equally well and significantly outperformed unnormalized words.

2.2 N-gram Stemming

In their article on n-gram indexing, McNamee and Mayfield left unaddressed the question of whether n-grams achieved higher performance over stemmed words. But in other work they did a direct comparison of n-grams and stems and found that 4-grams performed on par with Porter's SNOWBALL stemmer and in three of eight CLEF 2002 languages performed significantly better. The tendency was that n-grams held an advantage in the more morphologically complex languages (i.e., Finnish, Swedish, German). Still, the question remained as to whether the advantage in accuracy was worth the degradation in run-time performance.

The key factor in increased disk space and run-times is the fact that each word generates multiple n-grams. If there were only some way to prune away most of the n-grams and yet preserve enough of the benefits of n-grams to still confer an advantage. The selection method for each word must be efficient as this is an operation that will be performed on every word in the corpus – billions of operations will be required. Mayfield and McNamee [2] introduced n-gram stemming where a single n-gram would be used to represent each word. The selected n-gram was chosen based on the relative document frequencies of the n-grams spanning the word under consideration. The least frequent n-gram was used; this makes sense if the hypothesis that frequent n-grams are more likely to be part of the morphologically variable part of a word, not the root form. As an example, consider the words *jugglers* and *juggling.* Using statistics from 110282 newspaper articles the constituent n-grams of each word are shown in Table 1. It can be clearly seen that the rarest n-gram 'jugg' occurs much less often than the suffixes 'ers_' and 'ing_'. And both words would get replaced with 'jugg' which intuitively seems like a good choice.

Other methods for selecting a single n-gram or a small number of n-grams are possible. For example, the examples we have given use n=4, but since three runs were permitted in this year's evaluation we also created lists using n=3 and n=5. One could consider variants such as using two disjoint 3-grams or never ending an n-gram on a vowel, or any number of other permutations that might have linguistic merit.

[1] The # symbol is used to denote whitespace.

[2] See http://en.wikipedia.org/wiki/Johns_Hopkins for an explanation of this unusual given name.

Table 1: Frequency of n-grams in *jugglers and juggling*

n-gram	df(jugglers)	n-gram	df(juggling)
#jug	681	#jug	681
jugg	**495**	jugg	**495**
uggl	6775	uggl	6775
ggle	5377	ggli	3003
gler	890	glin	4567
lers	9650	ling	55210
ers#	89701	ing#	106463

Table 2: Efficacy of N-gram Stemming and Full N-gram Indexing (CLEF)

	words	4-stems	4-grams
Bulgarian	0.2096	0.2583 (+23.2%)	0.2928 (+39.7%)
English	0.3897	0.3796 (-2.6%)	0.4099 (+5.2%)
Finnish	0.2278	0.2532 (+11.1%)	0.3754 (+64.8%)
German	0.3335	0.3404 (+2.1%)	0.3994 (+19.8%)
Hungarian	0.1983	0.2939 (+48.2%)	0.3568 (+80.0%)

3 Evaluation

3.1 Task 1: Morphology Induction

For Task 1 we simply reported, for each word, the 3, 4, or 5-gram with lowest collection frequency based on the training data provided for each language. We did not predict any attributes such as +PL or +3_PER. Since we did not set out to truly learn morphological rules, we are not really surprised by our results. Our runs had lower F-score than all other runs in each of the four languages. We note that our runs with 5-grams did have the highest precision in each of the four languages with with very low recall.

3.2 Prior Results in Information Retrieval

We have experimented with n-gram morphemes in previous CLEF datasets. In Table 2 we present results in five languages using data from CLEF 2002 (for English, German, and Finnish) and CLEF 2005 (Bulgarian and Hungarian) that illustrate the relative effectiveness of n-gram indexing and n-gram analysis compared to processing unnormalized words. We report performance using mean average precision. Comparisons should not be made across languages in Table 2, but the results are meaningful for the different conditions in each language.

N-gram stemming usually helps, though performance was slightly worse for English. The largest gains are in the more complex languages and n-gram stemming does not perform as well as the use of ordinary n-gram indexing where the set of all n-grams is used instead of a select one.

3.3 Task 2: Information Retrieval

In this competition the organizers took lists of normalized word forms and replaced words in the CLEF source documents with a canonical form. The test sets used were: CLEF-2005 (English), CLEF-2004 (Finnish), and CLEF-2003 (German). The Lemur retrieval engine was used with TF/IDF or Okapi BM25 term weighting. No automated relevance feedback was performed on queries.

Table 3: Task 2 Results for N-gram Stemming and Reference Conditions

	English-05	**Finnish-04**	**German-03**
No Analysis (dummy)	0.2783	0.3496	0.3559
Porter Stemmer	**0.3052**	0.3725	0.3566
4-Stems	0.2838	0.3049	0.3257
5-Stems	0.2885	0.3327	0.3618
Baseline Morfessor	0.2863	**0.3874**	**0.4105**
Gold Standard	0.2602	0.3128	0.3734

Table 4: Results from Prior CLEF evaluations

	English-05	**Finnish-04**	**German-03**
All 4-grams	0.3692	0.5064	0.5056
All 5-grams	0.3873	0.5236	0.4869

In Table 3 we compare our performance using 4-stems and 5-stems to several reference conditions. These figures are all based on the *withnew* condition, the scenario where all words in the IR collection were transformed, and with TF/IDF weighting.

The 4-Stems do not score as high as the 5-Stems. The 5-Stems outperform the dummy condition and the gold standard analysis in two out of three languages. The Morfessor baseline roundly beats 5-Stems in Finnish and German, though the two methods are comparable in English. The results in Table 2, using CLEF-2002 data (which was not used in MorphoChallenge 2007), also show little difference between the dummy condition (labeled 'words' in Table 2) and n-gram morphemes. However, in Table 2, the use of all n-grams that span a word yields sizable improvements. This is also the case on the test sets used in this year's competition.

In Table 4 we list results obtained using the JHU/APL HAIRCUT retrieval engine in the CLEF evaluations in 2003-2005. These runs also made use of relevance feedback, and are not directly comparable to the results in Table 3. Nonetheless these figures suggest the use of all n-grams yields better performance compared to a single selected one.

4 Summary

We set out to participate in the information retrieval evaluation, not the morphology induction task. Our method for producing morphemes is unorthodox, but we believe that it is suitable for use in information retrieval. We provided results based on our own information retrieval experiments that support this claim, and that offer a means to obtain improvement with n-grams without a run-time penalty. The technique is also well motivated when working in a language with no available morphological toolkit. The results from the Task 2 evaluation suggest that n-gram morphemes perform better than unnormalized words and similarly to the gold standard analysis, but not as well as the Morfessor algorithm.

References

[1] Marc Damashek. Gauging simularity with n-grams: Language-independent categororization of text. *Science*, 267:843–848, 10 February 1995.

[2] James Mayfield and Paul McNamee. Single n-gram stemming. In *SIGIR*, pages 415–416, 2003.

[3] Paul McNamee and James Mayfield. Character N-gram tokenization for european language text retrieval. *Inf. Retr*, 7(1-2):73–97, 2004.

Program Listing

```
#!/usr/bin/perl
# Take corpus of training data and an input wordlist, compute n-gram frequencies,
# and output transformed wordlist that reduce wordforms to the least frequent n-gram.
# Usage: CreateStems corpus.txt wordlist.txt

package main;

my %cnt = ();
my $nlen = 5;
my $corpfile = $ARGV[0];
my $wordfile = $ARGV[1];
open FH, ”<$corpfile”;
while (<FH>) {
  $line = $_;
  chomp($line);
  next if ($line =˜ /ˆ0\t/);
  $line =˜ s/\d+\t//; # remove leading junk
  $line =˜ s/\s+/_/g; # replace space with underscores
  $line = ”_” . $line . ”_”; # padding
  # split line into n-grams
  my $lenlim = length($line) - $nlen + 1;
  for(my $i=0; $i<$lenlim; $i++) {
    my $tok = substr($line, $i, $nlen);
    # adjust counts for each observed n-gram
    $cnt{$tok}++;
  }
}
close FH;
open FH, ”<$wordfile”;
while (<FH>) {
  $line = $_;
  chomp($line);
  $line =˜ s/\d+\s+//; # remove leading junk (i.e., the frequency), keep the word
  my $origword = $line;
  $line =˜ s/\s+/_/g; # replace space with underscores
  $line = ”_” . $line . ”_”;
  # split line (really word) into n-grams
  my $lenlim = length($line) - $nlen + 1;
  my $besttok = ””;
  my $lowestfreq = 100000000;
  for(my $i=0; $i<$lenlim; $i++) {
    my $tok = substr($line, $i, $nlen);
    if (exists($cnt{$tok}) && $cnt{$tok} < $lowestfreq) {
      $besttok = $tok;
      $lowestfreq = $cnt{$tok};
    }
  }
  if ($besttok eq ””) {
    # Word wasn't in corpus, or was < 2 chars in length. Use the original word.
    $besttok = $origword;
  }
  # Now output word and stem
  print ”$origword\t$besttok\n”;
}
close FH;
1;
```