3

The DDI Approach to Morphology

KENNETH W. CHURCH

DDI stands for "Don't Do It." White (1980) used the acronym DDI in a delightful paper on garbage collection. He argued 25 years ago that the garbage collector should be run rarely – perhaps once a year – if at all. That's pretty much how many of us work these days. We buy enough memory so we don't have to clean up more than once a week. Rebooting is easier than garbage collecting. As for disks, many of us buy enough to last a couple of years without cleaning up (too much). And when it comes time to clean up, we buy a new machine with even more disk space.

Cleaning up isn't much fun. It would be nice if my son cleaned up his room more often (but then I set a lousy example). Technology isn't the problem or the solution. Even if I came home with a fancy new tool that did most of the work, I bet there would still be many things my son would rather do than clean up his room (if given the choice).

So, what does morphology have to do with garbage collection? White suggests that garbage collecting is harmful. We find that morphological inferences are dangerous. Simple morphological inferences are better than complex inferences. But even simple inferences are worse than none. Like cleaning up, we don't want to do any more morphology than we absolutely have to. Most of us would opt for DDI, if given the choice.

There are lots of programs out there that take out the garbage and decompose words. Some of these programs work remarkably well, for the most part. But, why run such programs, if we don't have to?

There are obvious morphological patterns that we would like our computer programs to take advantage of, but there are also many potential pitfalls. All inferences introduce certain errors, but some inferences are safer than others.

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

To err is human, but to really screw things up, you need a computer.

The argument for morphology comes down to a lack of choice. Do we have to run such programs? It is said that English has rather impoverished morphology compared to other languages such as, say, Finnish. Perhaps so. Thus, we'll use morphology programs for Finnish (because we have to, not because we want to), but for English, we'll use the DDI method, because we can.

However, I wonder if DDI might be an option, even for Finnish. Suppose we cached the more frequent words in the dictionary. How big would the cache have to be? If space is the problem, is morphology the best solution, or are there more effective compression techniques?

3.1 Morphology and the Bell Labs Text-to-Speech (TTS) Synthesizer

I started working on morphology as part of the Bell Labs text-to-speech (TTS) project. I wrote the letter to sound rules. They didn't work very well, maybe 80 percent of the words were ok.

I knew the letter-to-sound rules weren't very good. Rather than fix them, I pushed for the DDI method (dictionary lookup). It wasn't that hard to raise the funds to buy the rights to a dictionary, but there was serious pushback on the memory. It was hard, in those days, to justify an extra 1/4 megabyte of memory.

Dictionary lookup had obvious advantages in terms of precision, compared to letter-to-sound rules. (Fewer inferences/guesses are better than more inferences/guesses.) However, to improve recall, the dictionary had to be extended with (morphological) inferences that are safer than letter to sound rules, but not as safe as DDI (dictionary lookup). In Coker et al. (1990), we evaluated a number of inference methods:

- Stress-Neutral Suffixes (including regular inflection): abandons = abandon + s, abandoning = abandon + ing, abandonment = abandon + ment, Abbotts = Abbott + s, Abelson = Abel + son.
- Primary-Stress Ending: addressee = address + ee, accountability = account + ability, adaptation = adapt + ation.
- *ity*-Class Ending: abnormality = abnormal + ity, Adomovich = Adam + ovich, Abmbrosian = Ambrose + ian.
- *al*-Class Ending: accidental = accident + al, combative = combat + ive.
- Suffix Exchange: nominee = nominate − ate + ee, Agnano = Agnelli − elli + ano, Bierstade = Bierbaum − baum + stadt.
- Prefix: adjoin = ad + join, cardiovascular = cardio + vascular, O'brien = O' + brien, Macdonald = Mac + donald.

- Compound: airfield = air + field, anchorwoman = anchor + woman, Abdulhussein = Abdul + hussein, Baumgaertner = Baum + gaertner.
- Rhyming: Plotsky (from Trotsky), Alifano (from Califano).

Tables 1 and 2 report coverage by token over two data sources:

- 1988 Associated Press (AP) newswire, plus
- a list of names from Donnelley Marketing.

Names are distinguished from non-names (general vocabulary) because names are relatively hard. A high-quality commercial dictionary was used for general vocabulary. In addition, there was a special purpose dictionary of 50,000 common American surnames.

Table 2 reports precision by morphological method. A single judge listened to approximately 100 names in each row and labeled them as:

- Good: That's how I would have said it.
- OK: I could imagine someone else saying it that way or I don't know.
- Poor: I know that's wrong.

Don't make an error-prone inference if you don't have to. Even DDI (dictionary lookup) is far from perfect. The judge labeled 2 percent of names in the surname dictionary as "poor."

If you have to make a morphological inference, focus on simple, safe inferences with high precision and recall. The stress-neutral case, which includes regular inflection, is simpler than many, but even so, the judge labeled 4 percent of them as "poor," twice as many as DDI.

In addition to morphological inferences, we also evaluated the rhyming inference, proposed by Byrd and Chodorow (1985). Rhyming is riskier than DDI and conservative morphological processes (such as regular inflection), but safer than aggressive morphology such as compounding. Compounding is nearly as risky as letter to sound rules (only 83 percent good). And that is about as bad as it gets.

3.2 Information Retrieval

Information Retrieval was one of the first fields to question the value of morphological inferences. Do stemmers help retrieval performance? See Table 8.1 in Frakes and Baeza-Yates (1992) for a summary of stemming experiments, many of which failed to find much of a difference in terms of precision and recall. Salton and Lesk (1968) conclude, "For none of the collections is the improvement of one method over the other really dramatic, so that in practice either procedure might reasonably be used."

Such a mixed bag of mostly negative results ought to be disturbing for those of us working in natural language processing. If it is hard to show that something as simple as stemming is helpful, how can we possibly justify our

TABLE 1 Simple inferences cover much of the Associated Press news.

Inference Method	Ordinary Words (Non-names)	Capitalized Words (Names)
Direct Hit (DDI)	75%	70%
Stress Neutral	17%	14%
ity-class	1%	2%
al-class	1%	1%
Rhyme	0%	2%
Prefix	2%	0%
Compound	1%	1%
Combinations	4%	8%
All Dictionary-Based Methods	100%	97%
Letter to Sound Rules	0%	3%

TABLE 2 Simple inferences are relatively reliable (on names in Donnelly Marketing List). Rows are sorted by the "Poor" column.

Method	Good	OK	Poor	Coverage
Direct Hit (DDI)	95%	3%	2%	60%
Suffix Exchange	93%	5%	3%	1%
Stress Neutral	91%	6%	4%	25%
Rhyme	88%	8%	4%	2%
ity-Class	91%	3%	6%	1%
al-Class	87%	8%	6%	1%
Compound	83%	3%	14%	2%
All Dictionary-Based Methods				98%
Letter to Sound Rules				2%

interests in more challenging forms of natural language processing such as part of speech tagging, word sense disambiguation and parsing?

In Church (1995), I argued that morphological variants like "hostage" and "hostages" should not be treated as one term, or two, but somewhere in between, perhaps a term and a half. I looked at the distribution of terms across documents in the AP newswire, as illustrated in Table 3. In the 1988 Associated Press Newswire, there were 619 documents that mentioned both "hostage" and "hostages," 479 documents that mentioned the former but not the latter, 648 that mentioned the latter but not the former, and 78,223 that mentioned neither. One can measure similarity statistics based on such contingency tables. The correlation of documents that mentioned "hostage" and documents that mention "hostages" is about $\frac{1}{2}$, well above 0, but well below 1.

Treating "hostages" and "hostage" as one term makes sense, under the

TABLE 3 Contingency Table (1988 Associated Press Newswire)

	hostages	¬hostages
hostage	619	479
¬hostage	648	78,223

vector space model, if the correlation is 1. Treating them as two terms would be appropriate if the correlation were 0. But the correlation is in between. The mixed bag of results arises, I suspect, because neither the one term simplification, nor the two term simplification, fits the data. In the spirit of "do no harm," I lean toward DDI (two terms), rather than conflate things that shouldn't be conflated.

There are some very interesting lexical patterns to these correlations. Some words have large correlations with their variants, and some don't. Nouns and words with "lots of content" (better keywords for information retrieval) tend to have higher correlations with their variant forms than function words and non-nouns with relatively little meaning.

- large correlations: hostage(s), reactor(s), rebel(s), guerrilla(s), abortion(s), delegate(s), drug(s), stock(s), pesticides(s), airline(s).
- small correlations: await(s), ground(s), possession(s), boast(s), belonging(s), compare(s), direct(s), shield(s), last(s), urge(s).

The correlations are remarkably stable over data collections. If a pair has a large correlation in one corpus, then the correlation tends to be large in other corpora. In Church (1995), I studied correlations of correlations across five years of the AP newswire for 999 pairs of words like "hostage" and "hostages" that differ by a final "s." If we know the correlation of these 999 pairs in one year of the AP, then we can account for 80 percent or more of the variance in predicting the correlations of these 999 pairs in another year.

Morphology is not that different from case normalization and other text normalization steps that are commonly used in practice, but may do more harm than good. Some pairs like *Hurricane*/*hurricane* have large correlations while others like *Continental*/*continental* and *Anytime*/*anytime* do not. The correlations are large when both members of the pair have a lot of meaning and they refer to the same thing (*Hurricane*/*hurricane*); the correlations are small when they refer to different things (*Continental*/*continental*)[1] or not much of anything (*Anytime*/*anytime*).

- Large correlations (between upper and lower case variants): Hurricane, Pope, Emperor, Lottery, Zoo, Ballet, Golf, Canal, Immigration.

[1] *Continental* is an airline; *continental* is not.

- Small correlations (between upper and lower case variants): Troy, Path, Editions, Continental, Burns, Levy, Haven, Rush, Anytime

Morphology and case normalization are similar to priming and repetition. If a document mentions a word once, then it is likely that we will see that word (and its friends) again in the same document, especially if the word is a good keyword with lots of meaning. In Church (2000), I looked at the distribution of "Noriega" across documents. Noriega was mentioned in quite a number of AP news articles (about 6 articles per thousand) when the US invaded Panama. Under standard independence assumptions, the chance of two Noriega's should be $(6/1000)^2$. We shouldn't expect to find an article with two or more Noriega's, but we have lots of them. Of those documents that mention Noriega at least once, 75 percent mention Noriega a second time.

Repetition of a word (and its morphological variants and other friends) is more likely for good keywords with lots of content and less likely for function words that aren't very useful for information retrieval. In Church (2000), I showed that distinctive surnames are more likely to be repeated than ordinary first names:

- Distinctive Surnames: Noriega, Aristide, Escobar
- Ordinary First Names: John, George, Paul

There are lots of important linguistic generalizations that produce undeniable distributional patterns. Morphology is one of many such factors. The information retrieval community had hoped that they could use standard off-the-shelf morphology programs in straightforward ways to take advantage of morphological patterns to improve precision and recall. Such attempts have not worked out very well.

In the spirit of "there is no data like more data," it is probably wise not to collapse morphological variants, unless you run out of data, and have no choice. If we have enough data to compute the distribution of each form of each lemma separately, we might as well do so. As long as we have plenty of data, there is no need to introduce unnecessary assumptions like perfect correlation or total independence.

In recent years, with all the excitement about the web, there has been more talk about what we can do with all the data we have, and less talk about running out of data. Of course, we never have enough data, but right now, there is more interest in finding clever ways to take advantage of all the data we have, and less interest in finding clever smoothing techniques to compensate for all the data we don't have. Right now, the glass is half full, not half empty.

3.3 Part of Speech Tagging

There are, of course, many other applications for morphology programs including part of speech tagging and spelling correction. Statistical part of

speech taggers make use of two sets of probabilities:

- Lexicon: $Pr(tag|word)$
- Context: $Pr(tag|context)$

Much of the literature has tended to concentrate on the context model, but that's the easy part. Typically there are only P^3 parameters in the context model, where P is the number of parts of speech, typically several dozen. The lexicon is far more challenging, since there are more parameters: $V \times P$, where V is the size of the vocabulary, typically between 10^5 and 10^6. V is typically much larger than P^2.

To illustrate the challenge with lexical probabilities, my favorite example is the word "yawn." "Yawn" occurs once in the training corpus (the Brown Corpus)[2] as a noun, and once as a verb. Based on this sparse evidence, we need to know, not only the probability that "yawn" could be a noun or a verb, but also the probability that it could be an adjective, or any other part of speech. Just because we haven't seen "yawn" as an adjective, doesn't mean it can't happen.

In fact, most words are like "yawn." Of the 50,000 words in the Brown Corpus, 80 percent appear 5 times or less. Given that we have more than 5 parts of speech, we have more parameters than data for most words. Perhaps the Brown Corpus is just too small, but if we collect a larger corpus, we'll discover a pile of new infrequent words. The more data we look at, the more we realize just how little we know.

One might hope that one could infer the lexical probabilities by reasoning across morphological variants, but I have never been able to make this work. In Church (1992), section 5.2, I looked at noun/verb ambiguous words like "yawn" and "yawns." Some of them are more likely to be nouns and some are more likely to be verbs. I was hoping that the probabilities of one morphological variant could be used to infer the probabilities for the other morphological variant, but I failed to find anything of use. While there are clear constraints on the grammatical possibilities, the statistical probabilities are less predictable.

Co-training (Blum and Mitchell, 1998), appears to be a more promising direction forward. That is, there are contexts where the tagger is likely to do relatively well. For example, after "the," we are much more likely to see a noun than a verb. It ought to be possible to run the tagger on vast quantities of untagged material and use observations based on this untagged data to estimate parameters that we couldn't estimate very well based on the relatively small amount of labeled training data that we happened to have, the one-million word tagged Brown Corpus.

[2] http://en.wikipedia.org/wiki/Brown_Corpus

Co-training needs to be performed carefully. Merialdo (1994) reported that performance degraded the more he iterated (unless he started with almost no labeled training data). Iterating always increases the model's likelihood scores, but doesn't always improve performance. Co-training should not be allowed to move the parameter settings very far from the estimates based on the labeled training data, especially when we have plenty of labeled training data for the parameter in question. With appropriate precautions, I believe, co-training will be more effective than morphological inference for estimating lexical probabilities.

3.4 Spelling Correction

These days, it is no longer as necessary as it used to be to use morphology and other tricks to reduce the size of the lexicon. McIlroy (1982) describes the heroics that were used in the original Unix Spell program to pack a small dictionary of 32,000 English words into a PDP-11 address space of 64k bytes. That's just two bytes per word.

Normally, hash tables use a hash function to jump quickly to the appropriate bucket, and then the key is checked more thoroughly against the keys in the bucket. But the Unix Spell program didn't have enough memory to store the keys in the table, so they didn't. It was necessary to introduce a lossy compression method that worked remarkably well in practice. Suppose we have a table of $N \approx 32,000$ words. Choose a prime, P, that is somewhat larger than N. P trades off compression for accuracy. Shrinking P saves memory, but introduces loss (false hits). Increasing P reduces loss, but consumes memory.

- Memory: $N[\frac{1}{log_e 2} + log_2 \frac{P}{N}]$ bits to store N words
- Loss: Pr(false hit) = $1 - (1 - 1/P)^N$

The Unix Spell program hashed each word in the dictionary and then mod the hash code by P, to obtain a number between 0 and $P-1$. If the hash function is chosen well, these numbers will have a Poisson distribution. Sort the hash codes and take first differences. These differences will be exponentially distributed with a mean of P/N, since the inter-arrivals of a Poisson process are exponential. The differences were then compressed using a Golomb code, an optimal Huffman code for exponentially distributed values.

During runtime, an input word would be hashed into a hash code from 0 to $P - 1$, using the procedure described above. This hash code would then be looked up in the compressed table (by performing the Golomb decode and computing running sums to invert the first differences). If the hash code was found in the table, then the program assumed the input word was correctly spelled. If not, the input word was flagged as potentially misspelled. There is a small probability, Pr(false hit), that an incorrectly spelled word would generate a false hit, hashing into the same place as some other correctly spelled

word, causing the Unix Spell program to fail to flag a misspelled word that it would have flagged, if not for the clever (but lossy) compression.

In addition to these heroic compression techniques, the Unix Spell program made heavy use of whatever tricks it could, including morphology; memory was unbelievably tight. According to McIlroy (1982), derived words were culled from the dictionary. Thus, "Wells" was culled because it could be analyzed as "Well" + "s." "Peters" was analyzed as "Peter" + "s." The culling was recursive, so "Peter" was also culled since it could be decomposed into "Pete" + "er"! Recursive application of such heuristics was risky but necessary, given the lack of memory.

These heroic compression methods are no longer necessary now that we have 32-64 bits of address space. Modern spelling correctors no longer need to make use of fancy (lossy) compression techniques. They no longer cull the dictionary as aggressively. Modern spelling correctors have larger dictionaries with 10^5 - 10^6 entries, many of which could be derived from other entries, but doing so would introduce (unnecessary) risk. Sometimes it is ok to add "er" to some other lexical entry, and sometimes it isn't.

3.5 Conclusion

There are lots of morphology programs out there, many of which work surprisingly well. Nevertheless, for many practical applications, we prefer not to use such programs, if we have the choice. Simple morphological inferences are better than complex inferences. But even simple inferences are worse than none.

References

Blum, Avrim and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT)*, pages 92–100. New York: ACM Press.

Byrd, Roy J. and Martin S. Chodorow. 1985. Using an on-line dictionary to find rhyming words and pronunciations for unknown words. In *Proceedings of the 23rd conference on Association for Computational Linguistics (ACL)*, pages 277–283.

Church, Kenneth. 1992. Current practice in part of speech tagging and suggestions for the future. In A. Mackie, T. McAuley, and C. Simmons, eds., *In Honor of Henry Kucera*, pages 13–48. University of Michigan: Michigan Slavic Studies.

Church, Kenneth. 1995. One term or two? In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 310–318. New York: ACM Press.

Church, Kenneth W. 2000. Empirical estimates of adaptation: The chance of two noriegas is closer to $p/2$ than p^2. In *Proceedings of the 17th conference on Computational linguistics (COLING)*, pages 180–186.

Coker, Cecil, Kenneth Church, and Mark Liberman. 1990. Morphology and rhyming: two powerful alternatives to letter-to-sound rules for speech synthesis. In *ESCA Workshop on Speech Synthesis*, pages 83–86.

Frakes, William B. and Ricardo Baeza-Yates, eds. 1992. *Information retrieval: data structures and algorithms*. Upper Saddle River, NJ: Prentice-Hall, Inc. ISBN 0-13-463837-9.

McIlroy, M. D. 1982. Development of a spelling list. *IEEE Transactions on Communications* 30:91–99.

Merialdo, Bernard. 1994. Tagging english text with a probabilistic model. *Computational Linguistics* 20(2):155–171.

Salton, G. and M. E. Lesk. 1968. Computer evaluation of indexing and text processing. *J. ACM* 15(1):8–36.

White, Jon L. 1980. Address/memory management for a gigantic lisp environment or, gc considered harmful. In *LFP '80: Proceedings of the 1980 ACM Conference on LISP and Functional Programming*, pages 119–127. New York: ACM Press.