

THE PRACTICAL VALUE OF N-GRAMS IN GENERATION

Irene Langkilde and Kevin Knight
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
ilangkil@isi.edu and knight@isi.edu

Abstract

We examine the practical synergy between symbolic and statistical language processing in a generator called Nitrogen. The analysis provides insight into the kinds of linguistic decisions that bigram frequency statistics can make, and how it improves scalability. We also discuss the limits of bigram statistical knowledge. We focus on specific examples of Nitrogen's output.

1 Introduction

Langkilde and Knight (1998) introduced Nitrogen, a system that implements a new style of generation in which corpus-based ngram statistics are used in place of deep, extensive symbolic knowledge to provide very large-scale generation (lexicons and knowledge bases on the order of 200,000 entities), and simultaneously simplify the input and improve robustness for sentence generation. Nitrogen's generation occurs in two stages, as shown in Figure 1. First the input is mapped to a *word lattice*, a compact representation of multiple generation possibilities. Then, a statistical extractor selects the most fluent path through the lattice.

The word lattice encodes alternative English expressions for the input when the symbolic knowledge is unavailable (whether from the input, or from the knowledge bases) for making realization decisions. The Nitrogen statistical extractor ranks these alternatives using bigram (adjacent word pairs) and unigram (single word) statistics collected from two years of the Wall Street Journal. The extraction algorithm is presented in (Knight and Hatzivassiloglou, 1995).

In essence, Nitrogen uses ngram statistics to robustly make a wide variety of decisions, from tense to word choice to syntactic subcategorization, that traditionally are handled either with defaults (e.g., assume present tense, use the alphabetically-first synonyms, use nominal arguments), explicit input specification, or by using deep, detailed knowledge bases.

However, in scaling up a generator system, these methods become unsatisfactory. Defaults are too rigid and limit quality; detailed input specs are difficult or complex to construct, or may be unavailable; and

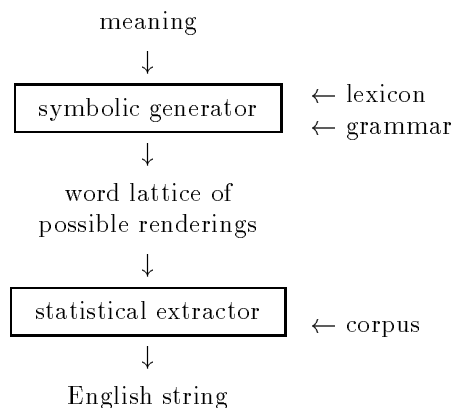


Figure 1: Combining Symbolic and Statistical Knowledge in a Natural Language Generator (Knight and Hatzivassiloglou, 1995).

large-scale comprehensive knowledge bases of detailed linguistic relations do not exist, and even those on a smaller scale tend to be brittle.

This paper examines the synergy between symbolic and statistical language processing and discusses Nitrogen’s performance in practice. The analysis provides insight into the kinds of linguistic decisions that bigram frequency statistics can make, and how it improves scalability. It also discusses the limits of bigram statistical knowledge. It is organized around specific examples of Nitrogen’s output.

2 Background

Nitrogen’s symbolic generator works from underspecified input and simple lexical, morphological and grammatical knowledge bases. Most of the linguistic decisions that need to be made in realizing the input are left up to the extractor. These include word choice, number, determinateness, subject-verb agreement, verb tense, passive versus active voice, verb sub-categorization, expression of possessiveness, and others. To illustrate this more concretely, we briefly describe Nitrogen’s symbolic knowledge bases and input.

Lexical knowledge in Nitrogen is stored as a list of mappings from words to conceptual meanings in the Sensus Ontology. The list is composed of tuples in the form:

```
(<word> <part-of-speech> <rank> <concept>)
```

Examples:

```
("eat"      VERB  1 |eat,take in|)
("eat"      VERB  2 |eat>eat lunch|)
("take in"  VERB 14 |eat,take in|)
```

The `<rank>` field orders the concepts by sense frequency for the given word, with a lower number signifying a more frequent sense. This simple lexicon contains no information about features like transitivity, sub-categorization, gradability (for adjectives), or countability (for nouns), etc.

Only the root forms of words are stored in the lexicon, so morphological variants must be derived. Nitrogen’s morphological knowledge base consists of a simple table that concisely merges both rules and exceptions. Here, for example, is a portion of the table for pluralizing nouns:

```
("-child" "children")
("-person" "people" "persons")
("-x" "xes" "xen")      ; boxes/oxen
```

Nitrogen’s list of exceptions is greatly reduced since the statistical extractor can be relied upon to discard non-word morphological derivations generated by the simplified rules (such as “boxen” or “oxes” generated from the rule above). These words never occur in the corpus, and so are assigned a very low score in the extractor phase.

Nitrogen’s grammar rules are organized around abstract semantic relations. This allows the details about syntactic structure to be decided internally, rather than requiring a client application to supply them. Nitrogen uses a *recasting* mechanism to reformulate the input in terms of relations that are less abstract, more syntactic, and closer to the final linear sentence realization. Recasting usually performs a one-to-many mapping (narrowing the final output to a single result is again left up to the statistical extractor). The multiple mappings reflect different constituent orders, different syntactic realizations, and different linguistic decisions for insertion of non-content words. Examples of the types of relations that Nitrogen handles are:

```
Semantic-- :agent, :patient, :domain, :range, :source, :destination, :spatial-locating,
           :temporal-locating, :accompanier;
Deep Syntactic-- :oblique1, :oblique2, :oblique3, :adjunct;
Shallow Syntactic-- :subject, :object, :mod, :tense, :quant, :definiteness, etc.
```

An example of Nitrogen’s input is the following:

```
(A / |workable|
  :DOMAIN (A2 / |sell<cozen|
            :AGENT I
            :PATIENT (T / |trust,reliance|
                      :GPI THEY))
  :POLARITY NEGATIVE)
```

This input is a labeled directed graph, or feature structure, that encodes relationships between entities A, A2, and T. Concept names are enclosed in vertical bars; we use an automated form of Wordnet 1.5 (Miller, 1990). The slash after a label is shorthand for the :INSTANCE relation.

Symbolic knowledge in Nitrogen is used to convert an input to a word lattice, from which the statistical extractor selects the best output. Nitrogen's extraction algorithm is based on a bigram scoring method where the probability of a sentence $w_1 \dots w_n$ is approximated as

$$P(w_1 | \text{START}) * P(w_2 | w_1) * \dots * P(w_N | w_{N-1}) * P(\text{END} | w_N).$$

This method is similar to the approach used in speech recognition. Every sentence path through the word lattice is ranked according to this formula, and the best one is selected as the final output.

2.1 Example 1

In order to describe the synergy that exists in Nitrogen between symbolic and statistical methods, it seems easiest to look in detail at few examples of Nitrogen's actual inputs and outputs. As a first example we will use the input above.

Symbolic generation for this input produces a word lattice containing 270 nodes, 592 arcs, and 155,764 distinct paths. Space permits only two portions (about one-fifth) of the sentence lattice to be shown in Figure 2. An example of a random path through the lattice not shown is "Betrayal of an trust of them by me is not having a feasibility." The top 10 paths selected by the statistical extractor are:

```
I cannot betray their trust .
I will not be able to betray their trust .
I am not able to betray their trust .
I are not able to betray their trust .
I is not able to betray their trust .
I cannot betray the trust of them .
I cannot betray trust of them .
I cannot betray a trust of them .
I cannot betray trusts of them .
I will not be able to betray the trust of them .
```

The statistical extractor inherently prefers common words and word combinations. When a subject and a verb are contiguous, it automatically prefers the verb conjugation that agrees with the subject. When a determiner and its head noun are contiguous, it automatically prefers the most grammatical combination (not "a trusts" or "an trust"). For example, here are the corpus counts for some of the subject-verb bigrams and determiner-noun bigrams:

I am	2797	a trust	394	an trust	0	the trust	1355
I are	47	a trusts	2	an trusts	0	the trusts	115
I is	14						

Note that the secondary preference for "I are" over "I is" makes sense for sentences like "John and I are..." Also note the apparent preference for the singular form of "trust" over the plural form, a subtle reflection of the most common meaning of the word "trust". This same preference is reflected for the bigrams "their trust" versus "their trusts."

their trust	28	their trusts	8
-------------	----	--------------	---

A purely symbolic generator would need a lot of deep, handcrafted knowledge to infer that the “reliance” meaning of trust must be singular. (The plural form is used in talking about monetary trusts, or as a verb). Our generator handles it automatically, based simply on the evidence of colloquial frequency.

A heuristic for preferring shorter phrases accounts for the preference of sentence 1 over sentence 2, and also for the preference of “their trust(s)” over “the trust(s) of them”. As can be seen from the lattice in Figure 2, the generator must also make a word choice decision in expressing the concept |**trust,reliance**|. It has two root choices, “trust” and “reliance.”

```
reliance 567      reliances 0      trust 6100      trusts 1083
```

The generator’s preference is predicted by these unigram counts. Trust(s) is much more common than reliance(s). Though one could modify the symbolic lexicon to suit their needs more precisely, doing so on a scale of 100,000 concepts is prohibitive. Thus the corpus statistics offer better scalability.

2.2 Example 2

Consider another input:

```
(A / |admire<look|
  :AGENT (V / |visitor|
    :AGENT-OF (C / |arrive,get|
      :DESTINATION (J / |Nihon|)))
  :PATIENT (M / "Mount Fuji"))
```

Lattice stats: 245 nodes, 659 arcs, 11,664,000 paths.

Random path: Visitant which came into the place where it will be Japanese has admired that there was Mount Fuji.

Top paths extracted:

```
Visitors who came in Japan admire Mount Fuji .
Visitors who came in Japan admires Mount Fuji .
Visitors who arrived in Japan admire Mount Fuji .
Visitors who arrived in Japan admires Mount Fuji .
Visitors who came to Japan admire Mount Fuji .
A visitor who came in Japan admire Mount Fuji .
The visitor who came in Japan admire Mount Fuji .
Visitors who came to Japan admires Mount Fuji .
A visitor who came in Japan admires Mount Fuji .
The visitor who came in Japan admires Mount Fuji .
Mount Fuji is admired by a visitor who came in Japan .
```

This example offers more word choice decisions to the generator than the previous example. There are two root choices for the concept |**visitor**|, “visitor” and “visitant,” and two for |**arrive,get**|. Between “visitor” and “visitant”, it is easy to guess that the generator will prefer “visitor.” However, the choice between singular and plural forms for “visitor” results in a decision opposite to the one made for “trust” above.

```
visitor 575      visitors 1083
```

In this case the generator prefers the plural form. In choosing between forms of “come” and “arrive,” the generator unsurprisingly prefers the more common word “come” and its derived forms.

For relative pronouns, the extractor is given a choice between “that,” “which,” and “who,” and nicely picks “who” in this case, though it has no symbolic knowledge of the grammatical constraint that “who” is only used to refer to people.

```
visitor_who 9      visitors_who 20
visitor_which 0    visitors_which 0
visitor_that 9     visitors_that 14
```


As can be seen from the lattice in Figure 3, the generator is given a wide variety of choices for verb tense. The heuristic preferring shorter phrases in effect narrows the choice down to the one-word expressions. Between these, the past tense is chosen, which is typical.

```
who_came 383      who_arrived 86
who_come 142      who_arrive  15
who_comes 89      who_arrives 11
```

A notable error of the generator in this example is in choosing the preposition “in” to precede “Japan.”

```
in_Japan 5413     to_Japan 1196
```

Apparently in general, Japan is usually preceded by the preposition “in.” It is the context of the verb “come” that makes one wish the generator would choose “to”, because “arrived in Japan” sounds fine, while “came in Japan” does not.

```
came_to  2443     arrived_in 544
came_in  1498     arrived_to 35
came_into 244     arrived_into 0
```

However, “in Japan” is so much stronger than “to Japan” when compared with “came to” versus “came in” that “in” wins. The problem here is that bigrams cannot capture dependencies that exist between more than two words. A look at trigrams (not used currently used by our system) shows that this dependency does indeed exist.

```
came_to_Japan  7      arrived_to_Japan  0
came_into_Japan 1      arrived_into_Japan 0
came_in_Japan  0      arrived_in_Japan  4
```

Later we will discuss possible improvements to Nitrogen that will address this problem.

A final aspect of this example worth discussing is the choice of person for the root verb “admire.” In this case, all the relevant bigrams are zero (ie. between “Japan” and “admire,” and between “admire” and “Mount,” so the decision essentially defaults to the unigrams.

```
admire 212      admired 211      admires 107
```

In this example the generator accidentally got lucky in its choice, because if there had been non-zero bigrams, they would have most likely caused the generator to make the wrong choice, by choosing a third person singular conjugation to agree with the contiguous word “Japan” rather than a third person plural conjugation to agree with the true subject “visitors”.

Note that another weakness of the current extractor in being able to choose the conjugation of any verb is that it weighs the bigrams between the verb and the word previous to it the same as it does the bigrams between the verb and the word that follows, although the former dependency is more important in choosing grammatical output. In other words, for a phrase like “visitors admire(s) Mount”, the bigrams “admire(s) Mount” are weighed equally with the bigrams “visitors admire(s)”, though obviously “Mount” should be less relevant in choosing between “admire” and “admires” than “visitors”.

3 Discussion

The strength of Nitrogen’s generation is in its simplicity and robustness. Basic unigram and bigram frequencies capture much of the linguistic information relevant to generation. Yet there are also inherent limitations. One is that dependencies between non-contiguous words cannot be captured, nor can dependencies between more than two items (in a bigram-based system).

As mentioned above, trigrams are one way of capturing part of these dependencies. However, they will only capture the contiguous ones, and moreover, practical experience suggests that switching to a trigram-based system might pose more problems than it solves. Trigrams exponentially increase the amount of data that must be stored, and require more extensive smoothing to prevent zero scores. Even in a bigram-based system the sparseness problem is very prominent. Many reasonable bigrams have zero raw counts, and many individual words may not appear in the corpus at all.

A more subtle disadvantage of trigrams is that many linguistic dependencies are only binary or unary relationships. A trigram-based system would not represent them efficiently; and due to sparseness, often not rank them correctly relative to each other. For example, the two phrases “An American admire(s) Mount Fuji” would not be correctly ranked by a trigram system since the relevant trigrams have raw zero counts, and would be smoothed relative to the unigram counts, which in this case favor “admire”, as shown above. However, the raw bigram counts are not all zero, and favor “admires”. So, although ternary relationships do need to be represented to improve the system’s quality, it seems better to selectively apply them instead.

In conjunction with this, we are developing a more detailed lattice representation to include part-of-speech and syntactic bracketing information. This will give us a handle for weighing certain ngrams more heavily (such as the one between a noun and its conjugated verb) and for representing non-contiguous relationships. Syntactic tags will alleviate the blurring problem that occurs when a lexical item has more than one part of speech, (e.g., in example 1 where the frequency count for “trusts” includes both its use as a noun and as a verb). We plan to use the Penn Treebank corpus (Marcus et al., 1994) in collecting this data.

Thus for now the symbolic half of Nitrogen’s generation mainly provides word order information and mappings from semantic relationships to general syntactic ones (like noun or verb phrases). Detailed syntactic information and agreement rules are omitted. It is still unclear how much more detail the symbolic rules need to encode. Long distance agreement is one area that our current system is weak in, but we expect to solve this problem statistically with more structured corpus data.

One important job for symbolic processing that we are working to improve is our system’s paraphrasing ability—its ability to express the same meaning with different syntactic constructions. This will facilitate the generation of longer, more complex meanings. For example an **agent-patient** input might need to be expressed as a noun phrase, rather than a sentence, to be more fluently expressed as the object of some external matrix verb.

Finally, a few rules of thumb we have learned: in collecting statistical data, it is necessary to distinguish proper nouns from each other, rather than lumping them all together. Otherwise it is impossible to prefer “in Japan” over “to Japan” because the only information available is “in PROPER” versus “to PROPER.” This also holds for numbers.

We also notice that some kind of length heuristic is necessary; otherwise the straightforward bigrams prefer sequences of simple words like “was”, “the”, “of”, to more concise renditions of a meaning. Yet our current heuristic sometime goes a little too far, dropping articles even when they are grammatically necessary. We expect the improvements we are working on will allow us to manage this problem better as well.

Bibliography

- K. Knight and V. Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proc. ACL*.
- I. Langkilde and K. Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proc. COLING-ACL*.
- M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.
- G. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4). (Special Issue).