# Instance-based Natural Language Generation

**Sebastian Varges** and **Chris Mellish**
LTG and ICCS
varges@cogsci.ed.ac.uk, chrism@dai.ed.ac.uk
Division of Informatics
University of Edinburgh

## Abstract

This paper presents a bottom-up generator that makes use of Information Retrieval techniques to rank potential generation candidates by comparing them to a data base of stored instances. We introduce two general techniques to address the search problem, expectation-driven search and dynamic grammar rule selection, and present the architecture of an implemented generation system called *IGEN*. Our approach uses a domain-specific generation grammar that is automatically derived from a semantically tagged treebank. We then evaluate the efficiency of our system.

## 1 Introduction

In recent years, Machine Learning techniques have been introduced into natural language generation (NLG) for at least three reasons: First, we often find an *interaction of decisions* at various levels and the need to deal with potentially conflicting constraints. These are typically difficult to handle in purely symbolic approaches. Next, the *knowledge acquisition bottleneck* makes it difficult to maintain and extend hand-crafted rule systems. This is especially true for problems like generating text in a specific style, for example. Furthermore, there is the problem of specifying the *generation input*: How can a client application possibly know about the feature values it has to select in order to drive a surface realizer?

The most prominent example of statistical NLG is *NITROGEN*, a sentence realizer designed for use in general-purpose machine translation (Langkilde and Knight, 1998). Further examples are the generation systems in (Ratnaparkhi, 2000) and (Bangalore and Rambow, 2000) as well as the use of statistical methods for more specific tasks, for example the ordering of NP premodifiers (Shaw and Hatzivassiloglou, 1999).

Instance-based Learning methods (IBL) (Aha et al., 1991) differ from statistical methods in that they are *lazy learning* approaches: they store previously encountered instances in memory and use them directly to process new input, rather than abstracting them into some statistical distribution. IBL is known to be able to learn exceptions in data well and adapt to subregularities. Since this is highly desirable for natural language processing in general, instance-based methods have been used in NLP under various names (*example*-based, *memory*-based, *case*-based; see Daelemans (1999) for an overview).

In this paper, we investigate the use of IBL-methods for natural language generation. In particular, we draw an analogy between IBL and Information Retrieval by noting that IR methods are essentially performing a form of nearest-neighbour search when computing the similarity between text documents. We believe that *instance-based natural language generation (IBNLG)* can make use of the experience in handling language data gained in the IR community. The so-called *bag-of-words* models of IR represent natural language texts as multisets of words without any linear order and use a distance metric such as the classical cosine distance to compute similarities between texts (Salton, 1989). Despite their simplicity these models have been highly successful in many document retrieval applications. In contrast to the *local* ngram models developed for speech recognition, for example, bag-of-words models reflect *global* word co-occurrence patterns. Since these bags are always treated separately for similarity computations, co-occurrence patterns in individual bags can be learned. Bag-of-words models therefore do not "abstract away" exceptions like statistical models do, and they are able to consider a larger context than many of these. Before one can do a comparison of IBNLG and statistical techniques for NLG, however, it is necessary to determine how IBNLG would work and indeed whether it can be made sufficiently efficient for practical use.

Like any other IBL-approach, instance-based generation has to answer the question how to represent and retrieve its stored instances and how to adapt them to process new inputs. Our proposal is to use standard IR techniques for representation and retrieval. This leaves us with the adaption task. We are basically subscribing to the overgeneration and ranking approach of *NITROGEN* but propose to tightly interleave rule-based generation and rank-

ing rather than employing two separate stages for these tasks. This amounts to doing the adaption before/during the computation of the nearest neighbour. Like other machine learning approaches to NLG, we are recasting an important part of the generation process in terms of disambiguation between alternatives. This is in contrast to the traditional view of generation as a deterministic, decision-making process.

The structure of the paper is as follows: in the next section, we give an overview of our system, followed by our approach to semantically annotating a domain corpus (section 3). The issue of automatic grammar construction is discussed in section 4. In section 5, we introduce the basic generation algorithm and concepts used for scoring candidates. In section 6, we present an expectation-driven ranking algorithm and in section 7 we show how grammar rules can be selected dynamically. For each of these we evaluate the efficiency of the ranker (section 8).

## 2 System Overview

We are interested in developing a corpus-based, domain-specific generation component that requires significantly less manual development effort than current approaches. The generator assumes that content determination has already taken place and the generation input has been chunked into sentence-sized units. This leaves some tasks of sentence planning, most notably lexical choice, and sentence realization to our system called *IGEN*. However, we cannot know whether the generator is actually able to express the entire input in a single sentence, and we expect our system to robustly handle gaps in expressibility.

Our domain of choice involves texts on management successions that can be found in the *Who's News* section of the *Wall Street Journal*. This domain has been used in the information extraction (IE) task of *MUC-6*. Our task is to generate the first sentences of these *Who's News* texts, effectively reversing the IE process. Following standard methodology we divide the corpus into training and test sets. The training set can then serve as an *instance base* for our IBNLG approach and we will use the test set to provide inputs to the system for evaluation. To this end, we add semantic markup to both training and test set articles.

The system consists of a *base generator* using a rule-based grammar to produce *candidates*, potential outputs of the system. The grammar is automatically derived from the semantically marked-up training set in combination with the corresponding Penn treebank structures. The *ranker* scores these candidates according to a similarity metric which measures their distance to the elements in the instance base. The ranks are determined by the similarity to the closest instances and the highest ranked sentence is chosen as the final generation output. *IGEN* uses standard chart generation techniques (Kay, 1996) in its base generator to efficiently produce generation candidates. The set of candidates forms a subset of the chart *edges* in that they comprise all edges of syntactic category *S*. The particular advantage of a bottom-up generation algorithm is that it allows our surface-based ranker to score edges as soon as they are built.

## 3 Semantic Annotation

The *MUC-6* IE task consisted of extracting information about incoming and outgoing person and the post name. Generating the actual texts requires considerably more input than what was extracted in the *MUC-6* task. We collected the first sentences of 142 *Who's News* articles of the Penn treebank II and developed a tagging scheme that also accounts for previous and other posts, dates, descriptions of companies and others. The overall number of different tags is 68. Examples are given in figure 1.

The input to the system is a set of tags ("slots") and fillers.[1] For most tags, the fillers are always of the same syntactic category, for example a proper name, a number expression or simply a noun. These can directly be rendered by the generator. However, in some cases, there is more variety concerning the category of the filler: for example, retirement as the reason for leaving a post is expressed as *'X, who is retiring'*, *'X, who retired'* or *'X will retire from his post as ...'*, (see tag [OUTEVENT_PRP_(SYNCAT) ...] in figure (1)). We added to these tags the syntactic category of the filler to allow them to be used only in the syntactically correct places by the rule system. This solves the practical problem but results in a potentially large number of unique tags and also gives the generator a strong hint towards a specific syntactic realization.

We divided the annotated corpus into a random selection of 105 articles for the training set and 37 for the test set. Within the training set, two sets of tags occurred 5 times, one set 4 times, 3 sets 3 times, 5 sets 2 times and 72 sets were singletons. There are just 6 bags of tags that occur in both training and test set.

The frequency distribution over individual tags is sharper: the most frequent tag in the training set is POST (138 occurrences), followed by INPERSON_FULLNAME and COMP_DESCR. There were only 33 singletons. The relatively low count of OUTPERSON_FULLNAME tags (14 occurrences) indicates that the articles tend to start with incoming events. Crucially, 5 types of tags only occur in the test set.

[1] Strictly speaking, we do not need the fillers until generation has finished but we use them to obtain a phonological string for each phrase (see below).

| tag | example | frequency (training set) |
|---|---|---|
| *INPERSON_FULLNAME* | [*INPERSON_FULLNAME* Gordon A. Paris] was elected ... | 104 |
| *OUTPERSON_FULLNAME* | , succeeding [*OUTPERSON_FULLNAME* David S. Black]. | 14 |
| *INPERSON_AGE* | [*INPERSON_AGE* 36] years old | 38 |
| *POST* | was named [*POST* director] | 138 |
| *INPERSON_PREVIOUSPOST* | formerly a [*INPERSON_PREVIOUSPOST* manager] | 16 |
| *COMP_DESCR* | of this [*COMP_DESCR* investment-banking company] | 88 |
| *BOARD_INCR* | , expanding the board to [*BOARD_INCR* 14] members. | 13 |
| *COMP_NATIONALITY* | [*COMP_NATIONALITY* US] [*COMP_DESCR* metals concern] | 0 |
| *OUTEVENT_DATE_EFFECTIVE* | , effective [*OUTEVENT_DATE_EFFECTIVE* Dec. 31]. | 5 |
| *POST_DESCR_(SYNCAT=NN)* | filling a [*POST_DESCR_NN* vacancy] | 3 |
| *OUTEVENT_PRP_(SYNCAT=VBD)* | , who [*OUTEVENT_PRP_VBD* resigned]. | 9 |

Figure 1: Example tags and phrases in management succession texts

```
****** marked-up sentence:
[INPERSON_FULLNAME Peter W. Likins], [INPERSON_OTHERPOST president] of [INPERSON_OTHERCOMP Lehigh University],
[INPERSON_OTHERCOMP_LOC Bethlehem, Pa.], was elected a [POST director] of this [COMP_DESCR maker of industrial
motion-control parts and systems].
***** input rules:
PP-COMP_DESCR <== of this COMP_DESCR   (f=72)
NP-POST <== a POST   (f=22)
NP-INPERSON_OTHERCOMP_LOC <== INPERSON_OTHERCOMP_LOC   (f=2)
NP-INPERSON_OTHERCOMP <== INPERSON_OTHERCOMP   (f=8)
NP-INPERSON_OTHERPOST <== INPERSON_OTHERPOST   (f=39)
NP-INPERSON_FULLNAME <== INPERSON_FULLNAME   (f=100)
***** phrase combining rules:
VP-POST <== was elected NP-POST PP-COMP_DESCR   (f=4)
PP-INPERSON_OTHERCOMP <== of NP-INPERSON_OTHERCOMP , NP-INPERSON_OTHERCOMP_LOC   (f=2)
NP-INPERSON_OTHERPOST <== NP-INPERSON_OTHERPOST PP-INPERSON_OTHERCOMP   (f=19)
NP-INPERSON_FULLNAME <== NP-INPERSON_FULLNAME , NP-INPERSON_OTHERPOST ,   (f=27)
S-INPERSON_FULLNAME <== NP-INPERSON_FULLNAME VP-POST .   (f=77)
```

Figure 2: cfg rules extracted for example sentence and their training set (instance base) frequency

## 4 Automatic Grammar Construction from Semantically Annotated Treebanks

After semantic markup has been added to the corpus, we use the syntactic treebank structure in combination with the additional markup to automatically produce a *semantic grammar*. The system needs to accept a set of semantic tags and generate initial phrases that can be combined to larger ones in a standard chart execution mechanism.

The basic idea is to identify portions of the tree structure around the semantic tags and allow these subtrees to be generated whenever the respective tag is available in the input. We are looking for the largest subtrees that do not overlap with other ones so that each corresponding *input rule* fires when a single tag is given. In the next step, we identify *phrasal rules* that combine several phrases and possibly some lexical material. The result of this process is the context-free backbone of a phrase-based generation grammar. There are thus two steps involved:

1. Input rule identification: recursively extend rule areas from input tags outwards, and stop extending areas when they are about to overlap with others. We only keep the mother node of the tree fragment, and ignore any internal structure. Furthermore, we omit any traces.
2. Phrasal rule identification: initially, phrasal rules are identified with input rule subtrees. These are then recursively extended - overlaps allowed - until the entire sentence structure is covered by a single rule.

To restrict rule combinations in a semantically oriented manner, we introduce new category labels for mother nodes of phrases that consist of the syntactic category obtained from the treebank plus the semantic tag of its leftmost daughter (figure 2).

This grammar construction algorithm has consequences for the semantic annotation step described above. In general, only constituents should be allowed to be marked up. In some situations however, Penn treebank words have to be broken in order to identify paths that will correspond to generation inputs (for example:`(JJ 54-year-old)`).

Words that are not supposed to be generated by a rule are not to be left untagged, even if we

are unsure about the actual tag, for example *'this company which also has an interest in book publishing'*. We tagged these pieces of text as ADDITIONAL_INFORMATION plus a more specific suffix referring to the type of information.

## 5 Basic Generation Algorithm

The grammar construction step outlined above extracts 328 rules (148 input rules and 180 phrasal rules) from the parse trees of the training set instances. The frequency distribution over these rules is very sharp again: the most frequent rule occurs 100 times and there are 235 singletons.

We automatically converted these rules into a notation that is suitable for use in our generation system. The rule system uses a production system based on the Rete algorithm (Forgy, 1982) to handle the relatively large number of rules. We assume a standard bottom-up chart algorithm (Gazdar and Mellish, 1989) which is initialized by adding edges for individual input tags to the agenda. For each new edge added from the agenda to the chart, all possible rule applications are computed to guarantee completeness of the chart. Grammar rules add new edges to the agenda and the algorithm stops when the agenda is empty. To guarantee semantic coherence, ie to avoid expressing some input tag more than once, a coherence-check is called for all non-first daughters of phrasal rules.

We rank new edges produced by the base generator according to their distance to the instances in the instance base before adding them to the agenda. An edge has as many cosine scores as there are instances and we use the *best* score to determine the agenda ordering. An example of a slightly simplified chart edge is shown below:

```
(VP-POST
   (phon <was elected a director>)
   (terms <was elected a POST>)
   (consumes {POST})     ; covered input tags
   (instances Ext-Addr) ; pointer to instance base
   (idx 67)             ; unique identifier
   (deriv <3>) (fired-by input-rule-11))
```

As in other IR and IBL applications, we need to decide on the *term representation* of instances and on the *weighting scheme* for these terms. Training set instances are generally represented as bags of terms of the form $I_i = \{t_{i,1}, ..., t_{i,n}\}$ where $t_{i,k}$ is the weight of term $k$ in instance $i$. The term representations contain the semantic tags and words outside tagged areas but not the fillers. For example, in a unigram model, the marked-up string *was elected a* [POST *president*] is represented as a document with weights for the terms *was*, *elected*, *a* and POST.

To be able to score edges, the ranker needs to obtain term representations of the edge contents that potentially match the *index terms* used for the representation of instances. The `terms` slot of the chart edges contains a list of surface elements, for example <was,elected,a,POST>. When edges are combined, these lists are concatenated and used to compute a *pseudo-document vector* similar to the one for instances. This allows us to experiment with ngram terms (like the bigrams {'was elected','elected a','a POST'}, for example) provided that the instances are represented similarly.

Terms are weighted by the well-known $tf \cdot idf$ term weighting scheme that assigns highest weights to terms that occur frequently ($tf$-component) in few instances ($idf$-component). Since instances and edges contain relatively few terms, $tf$ is less important than $idf$ which serves to discriminate between relevant and irrelevant "documents".

The definition of cosine distance between a candidate and an instance is

$$cos(\vec{c}, \vec{i}) = \frac{\sum_{j=1}^{n} w_{cj} \cdot w_{ij}}{\sqrt{\sum_{j=1}^{n} w_{cj}^2} \cdot \sqrt{\sum_{j=1}^{n} w_{ij}^2}}$$

where $w_{cj}$ is the weight of term $j$ for candidate $c$ and $w_{ij}$ the weight of term $j$ for instance $i$. We need normalization since otherwise ever more non-matching words could be added to an edge without penalties.

## 6 Expectation-driven Ranking

If we want to avoid the computational cost of computing the similarity between every new edge and the entire instance base, we have to consider two important characteristics of the search problem. First, edges have a number of cosine scores from different instances. If we want to use a Viterbi-style algorithm which defines equivalence classes for chart edges and only keeps the current best one for each class in a table, we need to maintain a separate table for each instance. This is because we cannot combine scores from different instances. In contrast, a single statistical model allows one to maintain a single table (Langkilde, 2000).

Second, the cosine score of an edge can increase as well as decrease when it is combined with other edges, depending on whether or not the newly added terms match the instance terms. Thus, the cosine score does not exhibit a monotonic behaviour that could be straightforwardly exploited.

We can, however, define the notion of the *expectation* of an edge that is monotonically decreasing. This is the potentially best cosine score an edge would have with respect to a specific instance if all missing matching words are added to it. To see its monotonic behaviour consider the following: The expectation of an initially empty edge $e$ is 1 since it contains no non-matching terms: $exp(e_{empty}, i) = 1$.

For each term $t$ that is added to $e$, two cases can arise:

1. $t$ does not occur in $i$. This will decrease the score as well as the expectation since it only increases the denominator of the cosine formula.
2. $t$ occurs in $i$. If $t$ has not been present in $e$ before, the expectation does not change since it already assumes that all correct words have been added. Otherwise, the score increases. If $t$ occurs in $e$ already, we have to consider the case of a $tf$-component in the edge term weights.[2] If the number of occurrences of $t$ in $e$ exceeds its number in $i$ ($tf_{t,e} > tf_{t,i}$), $e$ is able to neutralize the effect of its non-matching terms by repeating correct terms. Therefore, we impose an upper bound on $tf_{t,e}$ in the numerator (but not the denominator) of the cosine and restrict it to $tf_{t,i}$. As a result, exceeding correct terms in $e$ can only lead to an increase of the denominator and will decrease the cosine score.

A monotonically decreasing expectation thus requires two assumptions: First, edges are always extended monotonically. Terms are not allowed to be removed.[3] Second, it requires an upper bound on the $tf$-component of edge term weights. In practice, we do not find any noticible difference between using the normal cosine or the bounded one. We explain this partially by the effect of the coherence constraint in the base generator which prevents repeated realizations of the input semantics.

We can now modify the ranker to take advantage of the expectation. The main idea is that as soon as we have a first candidate and its *score*, we can discard all similarity computations that involve an *expectation* that is lower than this score. Thus, the score of already available candidates serves as a threshold $th$ that can be dynamically adjusted to the current best candidate as more candidates are being constructed.

For each chart edge $e$ we define the notion of *relevant instances* $ri_e$. This is a table of all instances $i$ for which the edge has an expectation higher than the score of the current best candidate. Each $i$ in $ri_e$ is associated with the current expectation: $ri_e = \{i_1 : exp(i_1, e), ..., i_n : exp(i_n, e)\}$. As long as no candidate is available, we assume a threshold of 0 to allow all instances to be considered. For each $e_{new}$ built by a grammar rule, the ranker performs the following steps:

1. Determine $ri_{e_{new}}$: For chart edges built from scratch, $ri_{e_{new}}$ contains references to the entire instance base with an expectation of 1: $ri_{e_{new}} = \{i_1 : 1, ..., i_n : 1\}$. For chart edges built by combining existing edges, say $e_1$ and $e_2$, a cheap initial approximation of $ri_{e_{new}}$ is obtained by intersecting $ri_{e_1}$ and $ri_{e_2}$ and taking the *lower* expectation of the remaining $i$. This is justified because the resulting edge will contain terms of both $e_1$ and $e_2$. Thus: $ri_{e_{new}} = ri_{e_1} \cap ri_{e_2}$ where for each $i \in ri_{e_{new}}$: $exp(i, e_{new}) = exp(i, e_1)$ if $exp(i, e_1) \leq exp(i, e_2)$ and $exp(i, e_{new}) = exp(i, e_2)$ otherwise.
2. Check the expectations in $ri_{e_{new}}$ against the current threshold $th$. For all $i \in ri_{e_{new}}$: if $exp(i, e_{new}) > th$[4] keep $i$ in $ri_{e_{new}}$, else remove $i$.
3. Compute the actual $exp(i, e_{new})$ for all $i \in ri_{e_{new}}$.
4. Add $e_{new}$ with its updated $ri_{e_{new}}$ to agenda unless $\mid ri \mid = 0$ in which case we can drop $e_{new}$.
5. If $e_{new}$ also qualifies as a candidate, compute $cos(e_{new}, i)$ for all $i \in ri_{e_{new}}$ and add it to the current list of candidates. If $cos_{max}(e_{new}) > th$ adjust threshold: $th = cos_{max}(e_{new})$.

The algorithm incrementally constrains the set of relevant instances $ri$. It has the advantage of being independent of the order in which chart edges are combined and it finds a single (or all) globally best solutions based on optimistic assumptions about best possible scores. It can deliver an output as soon as the first candidate is available, and further computation can be used to find potentially better ones. The performance of the expectation-based algorithm depends on how quickly a good candidate can be constructed since this sets the threshold for the remaining edges. If no candidate is available at all, no pruning can take place and the algorithm reverts to exhaustive search. However, these seem to be rather rare cases in practice (see section 8).

## 7 Dynamic Rule Selection

A further technique to increase the efficiency of the system is to reduce the number of grammar rules and instances before generation starts. We compute the cosine similarity between the semantic input and the instances represented by their semantic tags only. This only involves a single cycle over the instance base. We can then select the $n$-best instances for all further similarity computations. Furthermore, we can characterize each instance by a bitvector representation of the grammar rules derived from it in the grammar construction step. We determine the rules corresponding to the $m$ semantically closest instances by the union of their bitvector representations. Generation can then proceed as before with

[2] We assume that the same weighting scheme applies to both candidate and instance terms.

[3] In a unigram bag-of-words model, reordering of the surface string by grammatical operations would still be possible since this does not affect the term representation.

[4] We can use $exp(i, e_{new}) \geq th$ to find all best solutions.

```
**** chosen output candidate - correct:
1/258) Francis D John , president , was named to the additional post of chief executive officer .
       (exp:5:bigram, cos=1.0)
       (Input: INPERSON_FULLNAME: Francis D John,  INPERSON_AGE: 35,  INPERSON_OTHERPOST: president,
               POST_DESCR_JJ: additional,  POST: chief executive officer)
1/392) Robert D Paster , 49 years old , Space Shuttle Main Engine program manager , was named
       president of Rocketdyne division , a unit of this aerospace concern . (exhaust:5:unigram, cos=0.86)
**** chosen output candidate - minor errors:
1/19) Richard Schwartz resigned of an president of this aerospace concern . (exhaust:20:unigram, cos=0.74)
1/2469) Steven C Walker , senior vice president , was elected to the president of this bank
        holding company . (exhaust:20:unigram, cos=0.83)
**** chosen output candidate - semantic problems:
1/118) C Hyde Tucker was named chief executive officer and president of this company 's Bell Atlantic
       International Inc. (exp:105:unigram, cos=1.0)
**** some lower ranked candidates:
580/603) vice president , and a Space Shuttle Main Engine program manager was named to its president .
         (exp:105:unigram, cos=0.46)
1746/1780) John F Barrett , executive vice president , , 40 , was named chief operating officer , the
           vacant posts  (exp:105:unigram, cos=0.39)
589/676) the senior vice president was named a president of the bank holding company and the director .
         (exp:105:unigram, cos=0.43)
**** edge dropped by expectation-based ranker:
* Scott C Smith , formerly  former vice president finance, , formerly former chief financial officer
  were elected senior vice president of this media concern .
```

Figure 3: Rank and cosine score of generation candidates

a reduced number of instance tables and grammar rules from the beginning of processing.

## 8 Experiments

We tested our system with different settings by using the semantic tags of the test set as generation inputs. Although we focus on efficiency issues in this paper, we would like to give some example outputs produced by *IGEN* (figure 3). Different inputs and ranking models have been used (the notation for the system parameters in figure 3 is described below). Note for example, that the system is able to produce two uses of *named*. This is due to high similarity scores with respect to different nearest neighbours (figure 3, first and second sentence). We also give the semantic input for the first output sentence. In this case, the highest ranked candidate (of 258) expresses 4 out of 5 input tags.[5]

A preliminary evaluation[6] of the quality of the highest ranked sentences shows that 48.2% are entirely grammatical, 22.5% contain minor errors, 11.7% have problems that make them difficult to interpret and in 17.6% of the cases there was no or only fragmented output. Depending on the application, the first one or two categories could be acceptable as generation output, the latter two not. Typical minor errors involve punctuation, and also the use of filler words in incorrect contexts since these are not part of the term representation. Some correctable errors are the direct result of the semantic annotation. For example, *was elected to the president* (sentence 4 in figure 3) can be attributed to the following tagging in the original text where no more specific post was given: *was elected to the* [POST *board*].

[5]Lower ranked sentences can contain more semantics.

[6]Two human judges, results averaged over 3 test runs.

| ranker:$\lvert ri_{init}\rvert$:ngram | $\lvert\overline{ri}\rvert$ | $\overline{t}$ (secs) | $\lvert\overline{c}\rvert$ | no $c$ | $\lvert\overline{e}\rvert$ |
|---|---|---|---|---|---|
| exhaust:105:unigram | 105 | 92.6 | 701 | 0% | 1803 |
| exhaust:20:unigram | 20 | 32.6 | 458 | 2.7% | 739 |
| exhaust:5:unigram | 5 | 5.4 | 56 | 8.1% | 118 |
| exp:105:unigram | 53.3 | 18.8 | 137 | 0% | 495 |
| exp:20:unigram | 9.62 | 16 | 194 | 2.7% | 410 |
| exp:5:unigram | 3.7 | 6.5 | 32 | 8.1% | 86 |
| exp:105:bigram | 32.5 | 8.6 | 81 | 0% | 382 |
| exp:20:bigram | 9.98 | 8.7 | 231 | 2.7% | 401 |
| exp:5:bigram | 4.0 | 2.0 | 33 | 5.4% | 81 |

Figure 4: Average values for test set inputs

Figure (4) shows the performance of different ranking schemes. The system parameters are given in the form `ranker:nbest-instances:ngram` where the nbest instances are selected according to their semantic similarity with the input. The expectation-based ranker (`ranker=exp`) uses the expectation to determine the agenda ordering. The average instance table size $|\overline{ri}|$ per test input decreases for the expectation-based ranker. In contrast, exhaustive search (`ranker=exhaust`) always has to use the initial instance base $ri_{init}$. All experiments were run with a 2 minute timeout which mainly affects exhaustive search with a large instance base. Search also stopped as soon as a candidate with a cosine of 1.0 was found. Restricting $ri_{init}$ and the grammar from 105 to the 5/20 semantically closest in-

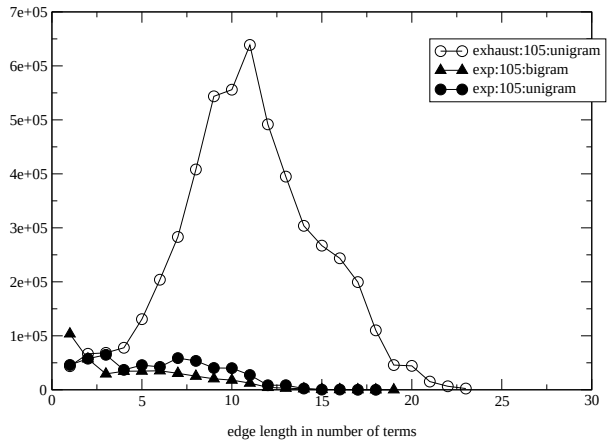


Figure 5: Overall number of similarity computations

stances reduces the average processing time $\overline{t}$. This also tends to reduce the average number of edges $|\overline{e}|$ and candidates $|\overline{c}|$ per test input at the risk of not finding a solution in some cases ('no $c$'). However, the interpretation of $|\overline{e}|$ and $|\overline{c}|$ is made difficult by the time limit and the stopping criterion.

For all similarity computations, we use $tf \cdot idf$ term weighting. We assign terms that do not occur in the training set the *idf* weight of a term that occurs in a single instance. This is especially relevant for semantic input tags that have not been seen in the training set. We also conducted experiments in which instances and edges are represented as bags of bigrams. This leads to a decrease in processing time which we explain by the fact that bigrams have a greater discriminative power according to the *idf* weighting scheme since they tend to be more rare than unigrams.[7]

Figure (5) shows the number of overall similarity computations for the 37 test inputs w.r.t. edge length as measured in number of terms. For exhaustive search, as shorter edges are combined into larger ones, there is an increase in the number of similarity computations, stopped only by the time limit and the size of the semantic input. In contrast, expectation-driven search manages to keep the number of similarity computations down.

## 9 Other Approaches

Woods (1982) introduces a shortfall scoring method for speech recognition that is similar in spirit to our expectation-driven ranking algorithm in that the search is guided by optimistic assumptions about the best possible score of parts of the input not yet covered by a partial solution. In contrast to our approach, (Woods, 1982) requires the allocation of best possible scores to parts of the input from the beginning of processing. Furthermore, scores need to be additive and the approach assumes that a lexical component enumerates words in decreasing order of score.

In contrast to *NITROGEN* but similar to Ratnaparkhi (2000), our system is domain-specific and makes crucial use of semantic information in the form of tags. *NITROGEN* also exhibits a bottom-up element in the way its lattice is constructed. However, this does not affect ranking and pruning which take place in a separate step. The architecture of Ratnaparkhi (2000) is different from both *IGEN* and *NITROGEN* in that it does not employ a grammar for candidate construction. In contrast, the overgeneration architecture of the latter two enables a *division of labour* to take place between rule-based and stochastic/instance-based knowledge sources. Although in our approach the training corpus serves the double purpose of providing the instance-base and the rule-based grammar, these components represent different aspects of the training set. Our approach works with many fewer examples for training than the one described in (Ratnaparkhi, 2000).

The example-based machine translation (EBMT) approach of (Sato and Nagao, 1990) uses a tree-based similarity metric that is used to first identify the closest instances and then adapt them. This is very different from the *surface-based* ranking scheme used in our overgeneration architecture. The EBMT system of (Brown, 1999) is more knowledge-poor than our approach in that new candidates are constructed by string/template matching and recombination rather than by means of a rule-based grammar.

(Miller et al., 2000) describe an approach to information extraction that also combines manual semantic annotations with syntactic structures. They use the resulting augmented trees to train a statistical model whereas we use them to extract a rule-based grammar.

## 10 Discussion and Future Work

Our algorithm is related to the class of $A^*$ algorithms in that it optimistically estimates the "cost" of reaching the goal from a given partial solution. This goal is defined as (partially) expressing the input semantics with the syntactic goal category. In the instance-based case however, we need to deal with a potentially large number of "cost estimates". We use existing solutions to reduce the search space for finding better ones by comparing scores and expectations across instances.

In this paper, we have concentrated on the efficiency of the basic generation algorithm, a fundamental issue in an overgeneration architecture. In future work, we would like to investigate the lin-

[7] We also experimented with other agenda orderings, higher order ngrams and different instance tables sizes which we cannot detail here.

guistic choices made in *IGEN* empirically, especially with respect to term weighting and representation. Errors in the output can give us hints as to whether more knowledge should be placed in the rule-based or the instance-based part of the system.

We currently consider all sentences as candidates and do not require semantic completeness since in general we cannot guarantee that the entire set of input tags is expressible in a single sentence. As a result, the different rankers express only 52-70% of the input tags in the best ranked sentence. This can be explained by the fact that the cosine similarity metric measures the distance to *any* instance regardless of the input semantics. In future work, we would like to adapt the ranking algorithm to be sensitive to the coverage of the input semantics.

We would also like to investigate the use of style in NLG. If we want a NLG system to learn the characteristics of a corpus of texts that exhibit different author's styles, it is usually more appropriate to generate a text that convincingly fits one *specific* style than one whose characteristics correspond to the average of all styles. An instance-based approach might be well-suited for this task.

## 11 Summary

To our knowledge, this paper presents the first incremental nearest-neighbour algorithm for NLG. We introduce the notion of *expectation* which allows us to define an algorithm that finds a globally optimal generation candidate since it never prunes promising branches of the search tree. The ranking algorithm can be used in combination with standard chart-generation techniques and does not make strong assumptions about the nature of the grammar used by the base generator. This effectively eliminates the need for the rule formalism to maintain special-purpose data structures for ranking.

We show how Information Retrieval methods can be used for NLG, opening the way for further use of IR techniques for generation. Our general approach requires a syntactic treebank of domain texts and - as the only manual effort - some semantic annotation. A particular advantage of the instance-based approach is its ability to work with very few example texts.

## Acknowledgements

This research has been supported in part by EPSRC and the German Academic Exchange Service (DAAD), program HSP-III.

The authors would like to thank Jon Oberlander, Chris Brew and the anonymous reviewers of this conference for valuable suggestions.

## References

David W. Aha, D. Kibler, and M. Albert. 1991. Instance-based Learning Agorithms. *Machine Learning*, 7:37–66.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a Probabilistic Hierarchical Model for Generation. In *Proceedings of COLING-00*, pages 42–48.

Ralf D. Brown. 1999. Adding Linguistic Knowledge to a Lexical Example-based Translation System. In *Proceedings of the Seventh International Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, pages 22–32.

Walter Daelemans. 1999. Memory-based Language Processing. Introduction to the Special Issue. *Journal of Experimental and Theoretical AI*, 11(3):287–467.

Charles L. Forgy. 1982. Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem. *Artificial Intelligence*, pages 17–37.

Gerald Gazdar and Chris Mellish. 1989. *Natural Language Processing in PROLOG. An Introduction to Computational Linguistics.* Addison-Wesley, Wokingham.

Martin Kay. 1996. Chart Generation. In *Proceedings of ACL-96*, pages 200–204.

Irene Langkilde and Kevin Knight. 1998. Generation that Exploits Corpus-based Statistical Knowledge. In *Proceedings of COLING/ACL-98*, pages 704–710, Montreal, Canada.

Irene Langkilde. 2000. Forest-based Statistical Sentence Generation. In *Proceedings of NAACL-00*, pages 170–177.

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A Novel Use of Statistical Parsing to Extract Information from Text. In *Proceedings of NAACL-00*, pages 226–233.

Adwait Ratnaparkhi. 2000. Trainable Methods for Surface Natural Language Generation. In *Proceedings of NAACL-00*, pages 194–201.

Gerald Salton. 1989. *Text Processing: the Transformation and Retrieval of Information by Computer.* Addison-Wesley, Ca.

Satoshi Sato and Makoto Nagao. 1990. Towards Memory-based Translation. In *Proceedings of COLING-90.*

James Shaw and Vasileios Hatzivassiloglou. 1999. Ordering among Premodifiers. In *Proceedings of ACL-99*, pages 135–143, University of Maryland, USA.

W. A. Woods. 1982. Optimal Search Strategies for Speech Understanding Control. *Artificial Intelligence*, pages 295–326.