Statistical Acquisition of Content Selection Rules for Natural Language Generation

Pablo A. Duboue and **Kathleen R. McKeown**

Department of Computer Science
Columbia University
{pablo,kathy}@cs.columbia.edu

Abstract

A Natural Language Generation system produces text using as input semantic data. One of its very first tasks is to decide which pieces of information to convey in the output. This task, called Content Selection, is quite domain dependent, requiring considerable re-engineering to transport the system from one scenario to another. In (Duboue and McKeown, 2003), we presented a method to acquire content selection rules automatically from a corpus of text and associated semantics. Our proposed technique was evaluated by comparing its output with information selected by human authors in unseen texts, where we were able to filter half the input data set without loss of recall. This report contains additional technical information about our system.

1 Introduction

CONTENT SELECTION is the task of choosing the right information to communicate in the output of a Natural Language Generation (NLG) system, given semantic input and a communicative goal. In general, Content Selection is a highly domain dependent task; new rules must be developed for each new domain, and typically this is done manually. Morevoer, it has been argued (Sripada et al., 2001) that Content Selection is the most important task from a user's standpoint (i.e., users may tolerate errors in wording, as long as the information being sought is present in the text).

Designing content selection rules manually is a tedious task. A realistic knowledge base contains a large amount of information that could potentially be included in a text and a designer must examine a sizable number of texts, produced in different situations, to determine the specific constraints for the selection of each piece of information.

Our goal is to develop a system that can automatically acquire constraints for the content selection task. Our algorithm uses the information we learned from a corpus of desired outputs for the system (i.e., human-produced text) aligned against related semantic data (i.e., the type of data the system will use as input). It produces constraints on every piece of the input where constraints dictate if it should appear in the output at all and if so, under what conditions. This process provides a filter on the information to be included in a text, identifying all information that is potentially relevant (previously termed *global focus* (McKeown, 1985) or *viewpoints* (Acker and Porter, 1994)). The resulting information can be later either further filtered, ordered and augmented by later stages in the generation pipeline (e.g., see the spreading activation algorithm used in ILEX (Cox et al., 1999)).

The research described here is part of the efforts for the automatic construction of the content planning module of PROGENIE, a biography generator (Duboue et al., 2003).

We focus on descriptive texts which realize a single, purely informative, communicative goal, as opposed to cases where more knowledge about speaker intentions are needed. In particular, we present experiments on biographical descriptions, where the

Actor, born Thomas Connery on August 25, 1930, in Fountainbridge, Edinburgh, Scotland, the son of a truck driver and charwoman. He has a brother, Neil, born in 1938. Connery dropped out of school at age fifteen to join the British Navy. Connery is best known for his portrayal of the suave, sophisticated British spy, James Bond, in the 1960s. ...

Figure 1: Sample Target Biography.

planned system will generate short paragraph length texts summarizing important facts about famous people. The kind of text that we aim to generate is shown in Figure 1. The rules that we aim to acquire will specify the kind of information that is typically included in any biography. In some cases, whether the information is included or not may be conditioned on the particular values of known facts (e.g., the *occupation* of the person being described —we may need different content selection rules for artists than politicians). To proceed with the experiments described here, we acquired a set of semantic information and related biographies from the Internet and used this corpus to learn Content Selection rules. The resource is introduced in Section 1.2; details of its construction are contained in Section 3.1.

Our main contribution is to analyze how variations in the data influence changes in the text. We perform such analysis by splitting the semantic input into clusters and then comparing the language models of the associated clusters induced in the text side (given the alignment between semantics and text in the corpus). By doing so, we gain insights on the relative importance of the different pieces of data and, thus, find out which data to include in the generated text.

The objective of this report is to explain at length the system and experiments summarized in (Duboue and McKeown, 2003) and it is divided into two parts: details regarding the algorithm (Section 2) and details regarding the experiments (Section 3).

1.1 Task and Methods

We thus work on the Content Selection task, defined as choosing the right information to communicate in a NLG system. An example is shown in Figure 2. Its input is a set of attribute-value pairs (Figure 2 (a)); and its output is a subset of the input attribute-value pairs (Figure 2 (b)).

We simplify Content Selection to better fit a learning approach. In particular, we focus on descriptive texts (single, informative, communicative goal) and the mining of high-level content selection rules, to filter out the input.

Our approach is thus the learning of content selection rules. The input to the system is a text and knowledge resource, or TKR (Figure 3 (a)). A TKR is a set of human-written text and knowledge base pairs. The knowledge base given as input will include the type of data that a purported generator will use to generate a text that satisfies the same pragmatic (i.e., communicative) goals being conveyed in the human input text. Nevertheless, it does not constitute an exact description of the semantics of the text. For our task at hand, a TKR constitutes *weak* evidence for learning, as more processing is required to determine whether the information appears in the text. Direct evidence for learning a Content Selection module is shown in Figure 3 (b), in the form of knowledge and labels (*selected* or *unselected*) over the knowledge.

The output of our system are Content Selection rules, constrained by what is in the data. We mine rules expressed in two different languages: the class-based rule language and the full-fledged content selection rule language. In the class-based language, the decision of whether to include each piece of information is made solely in their class, as determined by its path from the root of the semantic representation to the actual piece of data (*data path*). In the full fledged language, the values of all data in the knowledge base are used to decide whether or not to include each particular piece of it.

Central to our approach is the notion of *data paths* in the semantic network (an example is shown in Figure 4). Given a frame-based representation of knowledge, we need to identify particular pieces of knowledge inside the graph. We do so by selecting a particular frame as the *root* of the graph (the person whose biography we are generating, in our case, doubly circled in the figure) and considering the paths in the graph as identifiers for the different pieces of data. We call these *data paths*. Each path will identify a *class* of values, given the fact that some attributes are list-valued (e.g., the **relative** attribute in the figure). We use the notation

⟨`name first`⟩	John	⟨`name last`⟩	Doe
⟨`weight`⟩	150Kg	⟨`height`⟩	160cm
⟨`occupation`⟩	c-writer	⟨`occupation`⟩	c-producer
⟨`award title`⟩	BAFTA	⟨`award year`⟩	1999
⟨`relative type`⟩	c-grandson	⟨`rel. firstN`⟩	Dashiel
⟨`rel. lastN`⟩	Doe	⟨`rel. birthD`⟩	1990

(a)

⟨**`name first`**⟩	John	⟨**`name last`**⟩	Doe
⟨**`occupation`**⟩	c-writer	⟨**`occupation`**⟩	c-producer

(b)

John Doe is a writer, producer, . . .

(c)

Figure 2: Content Selection Example. (a) Input: Set of Attribute Value Pairs. (b) Output: Selected Attribute-Value Pairs. (c) Verbalization of the selected knowledge.

⟨`name first`⟩	John	⟨`name last`⟩	Doe
⟨`weight`⟩	150Kg	⟨`height`⟩	160cm

⇓

John Doe, American writer, born in Maryland in 1967, famous for his strong prose and sharp remarks,. . .

(a)

⟨`name first`⟩	John	⟨`name last`⟩	Doe
⟨`weight`⟩	150Kg	⟨`height`⟩	160cm

⇓

⟨**`name first`**⟩	**John**	⟨**`name last`**⟩	**Doe**
⟨`weight`⟩	150Kg	⟨`height`⟩	160cm

(b)

Figure 3: Input to our learning system. (a) Actual input, a set of associated knowledge base and text pairs (weak evidence for learning) (b) Fully supervised input (obtained afterwards as a byproduct of the processing done by our system).

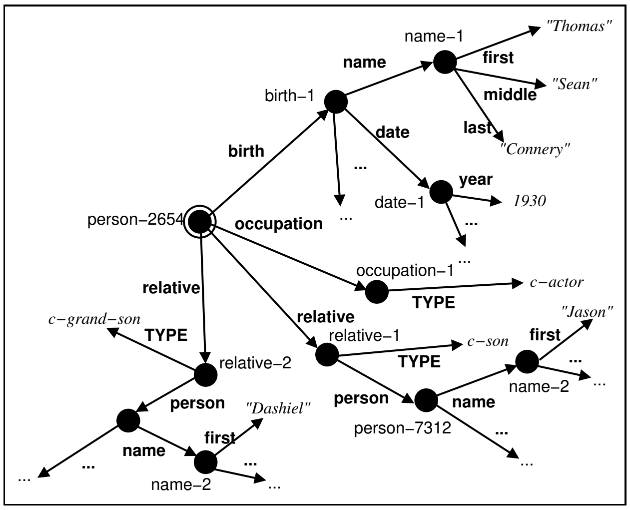

Figure 4: A frame-based knowledge representation, containing the triples (person-2654, **occupation**, occupation-1), (occupation-1, **TYPE**, *c-actor*), among others.

$\langle \texttt{attribute}_1\ \texttt{attribute}_2 \ldots \texttt{attribute}_n \rangle$ to denote data paths.

For the methods, we analyze how variation on the data influence variations in the text (we compare the cross entropy of clusters of text induced by clusters on the data). We basically try to extract enough information from our training material (Figure 3 (a)) to approximate the ideal training material (Figure 3 (b)) and then apply a standard machine learning apparatus.

1.2 Domain: Biographical Descriptions

We are developing ProGenIE, a biography generator, as part of the joint Columbia University—University of Colorado Open Question Answering project (AQUAINT). Our goal is to provide our final users with means to quickly and concisely communicate information about persons of interest. We plan to combine a generator with an agent-based infrastructure expecting to ultimately mix textual (such as existing biographies and news articles) as well as non-textual (such as airline passengers lists and bank records) sources. We will use the examples from the domain contained in the corpus described here to automatically construct content planning schemata. These schemata will guide the generation of biographies on unseen people.

Biography generation is an exciting field that has attracted practitioners of NLG in the past (Kim et al., 2002; Schiffman et al., 2001; Radev and McKeown, 1997; Teich and Bateman, 1994). It has the advantages of being a constrained domain amenable to current generation approaches, while at the same time offering more possibilities than many constrained domains, given the variety of styles that biographies exhibit, as well as the possibility for ultimately generating relatively long biographies.

The AQUAINT project focuses mostly on military and intelligence targets. However, there is a lack of publicly available information about such targets. Therefore, we shifted our attention to more popular individuals. As our approach is based on machine learning, given enough training data, the particular biographical field chosen is of immaterial importance.

By far the most popular domain for biographies and assorted data about people is the *celebrities* domain. Most fans are eager to express their support of their favorite actor, model, singer or director by collecting sizable amounts of trivia or assembling them in very detailed biographies. The availability of information in this domain has lured other researchers into working with it (Pang et al., 2002; Taskar et al., 2001).

We have gathered a resource of text and associated knowledge in the biography domain, as explained at length in Section 3.1. More specifically, our resource is a collection of human-produced texts together with the knowledge base a generation system might use as input for generation. The knowledge base contains many pieces of information related to the person the biography talks about (and that the system will use to generate that type of biography), not all of which necessarily will appear in the biography. That is, the associated knowledge base is not the semantics of the *target text* but the larger set[1] of all things that could possibly be said about the person in question. The intersection between the input knowledge base and the semantics of the target text is what we are interested in capturing by means of our statistical techniques.

[1]The semantics of the text normally contain information not present in our semantic input, although for the sake of Content Selection is better to consider it as a "smaller" set.

2 Algorithm

Figure 5 illustrates our two-step approach, divided in a number of modules. In the first step (Module (1) and output (A) in the Figure), we try to identify and solve the easy cases for Content Selection. The easy cases in our task are pieces of data that are copied verbatim from the input to the output. In biography generation, this includes names, dates of birth and the like. After these cases have been addressed, the remaining semantic data is clustered and the text corresponding to each cluster post-processed to measure degrees of influence for different semantic units. Further techniques are then employed to improve the precision of the algorithm.

At a finer level of detail, our Content Selection induction system can be divided into five parts (marked (1) to (5) in the Figure): Matching (Section 2.1), Clustering (Section 2.2), Selection (Section 2.3), Distillation (Section 2.4) and Extraction (Section 2.5). It produces three different outcomes (marked as (A), (B) and (C) in the figure), *baseline rules*, *class-based rules* and *content selection rules*.

2.1 Matching

In the first stage (cf. Figure 5(1)), the objective is to identify pieces from the input that are copied verbatim to the output. These types of verbatim-copied anchors are easy to identify and they allow us do two things before further analyzing the input data: remove this data from the input as it has already been selected for inclusion in the text and mark this piece of text as a part of the input, not as actual text.

The rest of the semantic input is either verbalized (e.g., by means of a verbalization rule of the form $\langle$`brother age`$\rangle < 35 \Rightarrow$ *"young"*) or not included at all. This situation is much more challenging and requires the use of our proposed statistical selection technique.

The matching module, then, takes as input a text and associated knowledge base from the TKR and searches for direct occurrences of atomic values in that knowledge base within the text. For example, given an attribute-value pair such as ($\langle$`name first`$\rangle$, *"John"*), we search for the string *"John"* in the text (e.g., *"An icon of American cowboy movies, **John** Wayne entertained several generations of Westerns fans…"*). We employ a variant of the scripts used to tokenize the PennTrebank[2] to tokenize both the full text and the verbalization of the atomic value. To search for the atomic value in the full text, we employ a naïve search method, $O(n^2)$.[3] Figure 6 shows an example of the obtained matched text.

The matching is done for each data path. Before the actual matching, data paths with exact repetitions are clustered into equivalence classes. These "repeated links" are paths that always mean the same (in general, as a result of artifacts of the knowledge representation), for example, the name of the mother and the name of the mother at the birth event. The process of finding the repeated links has three steps:

1. Hypothesis forming: record all pairs that appear repeated.

2. Hypothesis rejecting: check that every time the same pair appears, it is linked to the same object.

3. Hypothesis merging: use transitive relations to build the classes.

Finally, for the actual matching we take into account the following items:

- The matching is done over sequences of semantic tokens.

- Longer and earlier matches are preferred.

- Punctuation characters are disregarded.

The overall matching operation is prone to two type of errors: omission (if the atomic value is symbolic in nature, e.g., `c-young` or a different verbalization of the concept *MA* instead of *Maryland*) and overgeneration (if there are several values for it, e.g., *"John"* appears in the text and it is simultaneously the name of the father and a brother).

In the first case, omission errors, the system will employ the cross-entropy cluster-based method described in Section 2.3 to determine whether or not the information appears somehow in the text.

[2] `http://www.cis.upenn.edu/~treebank/tokenizer.sed`.

[3] In the case of natural language text the naïve method is comparable to more complex to implement methods, such as KMP or Booyer-Moore (Gusfield, 1997).

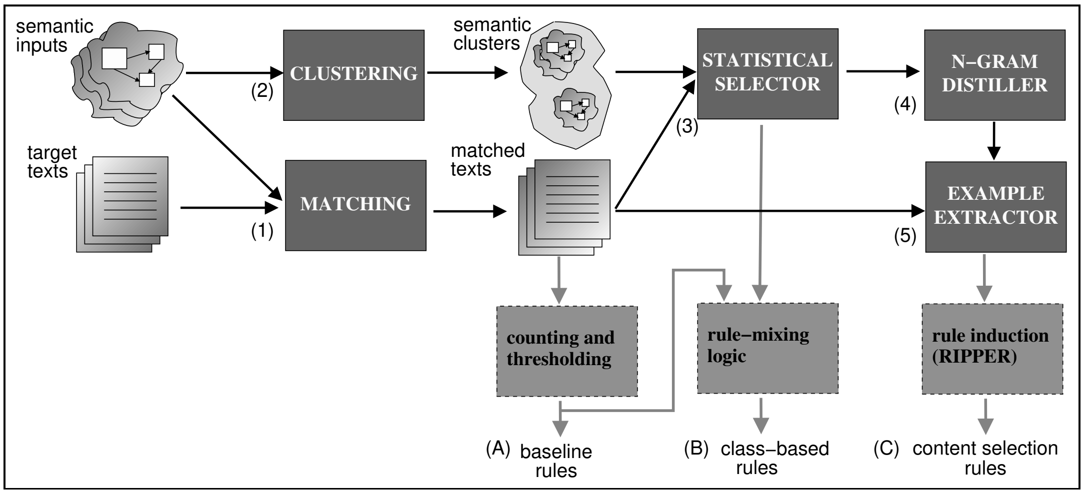

Figure 5: The rule induction system.

For the second problem, the system will do an educated guess, employing the priors of each data path over the whole corpus (normalized counts of the number of times a given data path appears defined in a given knowledge base).

2.2 Clustering

For each class in the semantic input that was not ruled out in the previous step (e.g., ⟨`brother age`⟩), we proceed to cluster (cf. Figure 5(2)) the possible values in the path, over all people (e.g., $[1 \leq age \leq 24]; [25 \leq age \leq 50]; [51 \leq age \leq 90]$ for age). In the case of free-text fields, the top level, most informative terms,[4] are picked and used for the clustering. For example, for *"played an insecure young resident"* it would be [*played*, *insecure*, *resident*].

Having done so, the texts associated with each cluster are used to derive language models (in our case we used bi-grams, so we count the bi-grams appearing in all the biographies for a given cluster —e.g., all the people with age between 25 and 50 years old, $[25 \leq age \leq 50]$).

We cluster the data according to its different values, to analyze how meaningful variations on the data side affect language models of the associated documents. This data clustering process is fully automated and totally data-driven. No prior knowledge about the data is used; in particular, we employ no additional ontological knowledge. That makes for a very difficult setting, although it strives for widespread applicability.

Two issues were important here: first, the obtained clusters have to be meaningful, that is, they have to represent natural associations within the data; and, second, each of the individual clusters has to be large enough to allow generating reasonable language models on the text side.

For the experiments described in the paper, we employed three clustering methods: a clustering with centroids and distance function, a random aggregation clustering and a clustering with a default set. We explain them in turn. Each of these methods normally have different strategies to deal with numeric and non-numeric (String) values. They are parameterized with the minimum size for each cluster (CLUSTER_THRESHOLD) and a dictionary containing the information content of words (IDF), measured in a corpus of domain-related text.

The first method is a classic agglomerative clustering with a distance. We keep a centroid for each cluster and iteratively join clusters until all of them have the minimum size. We compute the distance between the centroids of every pair of clusters and join the two with the minimum distance. Distances are computed over the numerical differences on numeric values and on the weigthed counts of overlapping words in string values (weighted with the information content, a vector representation).

[4] We use the maximum value of the TF*IDF weights for each term in the whole text collection. That has the immediate effect of disregarding stop words.

Ryder ⟨name last⟩ , Winona ⟨name first / birth place city / birth name first⟩ 1.0 1971 ⟨birth date year⟩ 1.0 –). Actress. Born

Winona ⟨name first / birth place city / birth name first⟩ 1.0 Laura ⟨birth name givenname⟩ 1.0 Horowitz ⟨birth father name last / birth name last / birth mother name last⟩ 0.5 ⟨relative relative name last⟩ 0.5 on

October ⟨birth date month⟩ 1.0 29 ⟨birth date day⟩ 1.0 . (1971 ⟨birth date year⟩ 1.0 in Winona ⟨name first / birth place city / birth name first⟩ 1.0

, Minnesota. Named after the city where she was born, she is the third of four siblings including one half brother and one half sister from her mother's first marriage. Ryder ⟨name last⟩ 's

parents Michael ⟨relative relative name first⟩ 0.5 ⟨birth father name first⟩ 0.5 and Cindy ⟨birth mother name first⟩ 0.5 ⟨relative relative name first⟩ 0.5 née Palmer

Horowitz ⟨birth father name last / birth name last / birth mother name last⟩ 0.5 ⟨relative relative name last⟩ 0.5 were hippie intellectuals and family friends included the likes of

beat poet Allen Ginsberg and counterculture guru Timothy Leary who was Ryder ⟨name last⟩ 's godfather.

Ryder ⟨name last⟩ 's family lived briefly in Colombia with Chilean revolutionaries before returning to northern California in 1974. Later the family moved to a commune in Mendocino where they lived for four years without television or electricity. They relocated to Petaluma California in the early 1980 s where Ryder ⟨name last⟩ attended school and developed an interest in dramatic arts. At the age of 12 her parents encouraged her to enroll in the American Conservatory Theater (ACT) in San Francisco ⟨education place city⟩ 1.0

. In 1985, Ryder ⟨name last⟩ was performing a monologue chosen from J. D. Salinger's Franny Zooey at ACT when Deborah Lucchesi, a talent scout, spotted her. Lucchesi arranged for Ryder ⟨name last⟩ to take a screen test for the upcoming Desert Bloom starring Jon Voight and Ellen Barkin. Ryder ⟨name last⟩ lost the part to Annabeth Gish but it wasn't long before she was cast in her debut role as Rina in David ⟨significant-other significant-other name first⟩ 1.0 Seltzer's coming of age film Lucas (1986). She shot the film during her summer vacation then entered eighth grade in the fall. She attended Petaluma High School where she graduated with a 4.0 grade point average (the highest possible score). Throughout her high school career however . . .

Figure 6: An example of a matched text (excerpt).

The last two methods are very related and we will discuss them together. They count the number of knowledge bases that contain each possible atomic data item (e.g., they start by counting how many knowledge bases contain *"December"* for ⟨`birth data month`⟩, and so on). If the number goes above CLUSTER_THRESHOLD, the cluster is left as-is (e.g., if there are six people born in December and CLUSTER_THRESHOLD is set to 5, then the six people stand as a cluster on their own). How to deal with small clusters is where the two methods differ. Clustering with random aggregation will aggregate smaller clusters picked at random until they result in a cluster of the minimum size. Clustering with default set will collapse all these smaller clusters in a large, default cluster. The rationale behind these two methods is that, without any prior or ontological knowledge, keeping each value separated from the rest is the best we can do.

2.3 Statistical selector

The statistical selection module deals with cases that are unmatched, by comparing the language models of the clusters of documents induced by the clusters of the data. What we want is to find a change in word choice correlated with a change in data. If there is no correlation, then the piece of data which changed should not be selected by Content Selection. We will first explain how the language models are built and then turn to the comparison between them.

We employed bigram models over sequences of tokens (marked text), where the sequences were previously filtered for stop words (controlled by the parameters USE_STOP_WORDS[5], FILTERED_WORDS[6], USE_DF_FILTERING[7] and STOP_WORDS_FILE[8]) and exact matched text was conflated to its class (understood as the path from the root of the knowledge base to the actual data instance). Given a set of such sequences of tokens, we computed bigrams by counting and normalization. No backoff was employed.

In order to compare language models, we turned to techniques from adaptive NLP (i.e., on the basis of genre and type distinctions) (Illouz, 2000). In particular, we employed the *cross entropy*[9] between two language models M_1 and M_2, defined as follows (where $P_M(m)$ is the probability that M assigns to the n-gram m):

$$CE(M_1, M_2) = -\sum_{n \in n-grams} P_{M_1}(n) \log P_{M_2}(n) \quad (1)$$

Smaller values of $CE(M_1, M_2)$ indicate that M_1 is more similar to M_2. On the other hand, if we take M_1 to be a model of randomly selected documents and M_2 a model of a subset of texts that are associated with the cluster, then a greater-than-chance CE value would be an indicator that the cluster in the semantic side is being correlated with changes in the text side.

We then need to perform a sampling process, in which we want to obtain CE values that would represent the null hypothesis in the domain. We sample two arbitrary subsets of k elements each from the total set of documents and compute the CE of their derived language models (these CE values constitute our control set). We then compare, again, a random sample of size k from the cluster against a random sample of size k from the difference between the whole collection and the cluster (these CE values constitute our experiment set). To see whether the values in the experiment set are larger (in a stochastic fashion) than the values in the control set, we employed the Mann-Whitney U test (Siegel and Castellan Jr., 1988) (cf. Figure 5(3)). We performed 20 rounds of sampling (with $k = 5$) and tested at the 0.05 significance level. Finally, if the cross-entropy values for the experiment set are larger than for the control set, we can infer that the values for that semantic cluster do influence the text. Thus, a positive U test for any data path was considered as an indicator that the data path should be selected.

Using simple thresholds and the U test, *class-based* content selection rules can be obtained. These rules will select or unselect each and every instance of a given data path at the same time (e.g., if ⟨`relative person name first`⟩ is selected, then

[5] Whether or not to use a stop words external file or to use dynamic filtering.

[6] In dynamic filtering, the number of words to filer.

[7] Whether to use DF or lowest TF*IDF to do the filtering.

[8] Stop words external file, three were provided: stopwords from PubMed; non-nouns, non-verbs and non-nouns, non-adjectives and non-verbs.

[9] Other metrics would have been possible, in as much as they measure the similarity between the two models.

both *"Dashiel"* and *"Jason"* will be selected in Figure 4). By counting the number of times a data path in the exact matching appears in the texts (above some fixed threshold) we can obtain **baseline** content selection rules (cf. Figure 5(A)). Adding our statistically selected (by means of the cross-entropy sampling and the U test) data paths to that set we obtain **class-based** content selection rules (cf. Figure 5(B)). By means of its simple algorithm, we expect these rules to overtly over-generate, but to achieve excellent coverage. These class-based rules are relevant to the KR concept of *Viewpoints* (Acker and Porter, 1994);[10] we extract a slice of the knowledge base that is relevant to the domain task at hand.

However, the expressivity of the class-based approach is plainly not enough to capture the idiosyncrasies of content selection: for example, it may be the case that children's names may be worth mentioning, while grand-children's names are not. That is, in Figure 4, $\langle$`relative person name first`$\rangle$ is dependent on $\langle$`relative TYPE`$\rangle$ and therefore, all the information in the current instance should be taken into account to decide whether a particular data path and it values should be included or not. Our approach so far simply determines that an attribute should always be included in a biography text. These examples illustrate that content selection rules should capture cases where an attribute should be included only under certain conditions; that is, only when other semantic attributes take on specific values.

To conclude, these issues were also important:

- Clusters with more that 80% of the instances were discarded.

- The Mann-Whitney U test has been tabulated for different numbers of samples. The largest tabulated number available in (Siegel and Castellan Jr., 1988) is 20. We thus decided to sample 20 times, to avoid having to approximate the test and being able to employ as much evidence as possible.

[10]they define them as *a coherent sub-graph of the knowledge base describing the structure and function of objects, the change made to objects by processes, and the temporal attributes and temporal decompositions of processes*.

2.4 n-gram distillation

The n-gram distillation module intends to obtain extra information from the algorithm described in the previous module. While the statistical selection module provides a gross estimate of the overall importance of a data path, it does not say whether or not a particular instance of a data path (e.g., (`prize`, *Oscar*) appeared in the associated text).

To obtain finer grained information, that should elucidate whether or not a piece of data appears in a given text-data pair, we turned to a *n-gram distillation* process (cf. Figure 5(4)), where the most significant n-grams (bi-grams in our case) were picked during the sampling process, by looking at their overall contribution to the CE term in Equation 1. For example, our system found the bi-grams $\boxed{\textit{screenwriter director}}$ and $\boxed{\textit{has screenwriter}}$[11] as relevant for the cluster ($\langle$`occupation TYPE`$\rangle$, `c-writer`), while the cluster ($\langle$`occupation TYPE`$\rangle$, $\{$`c-comedian`, `c-actor`$\}$) will not include those, but will include $\boxed{\textit{sitcom Time}}$ and $\boxed{\textit{Comedy Musical}}$. These n-grams thus indicate that the data path ($\langle$`occupation TYPE`$\rangle$, is included in the text; a change in value does affect the output. We later use the matching of these n-grams as an indicator of that particular instance as being selected in that document.

The distillation algorithm goes as follows: for each cluster, order the events (n-grams) by its $-P_{M_1}(n_{gram})\log(P_{M_2}(n_{gram}))$ value, keeping only the first NGRAMS_PER_DOC stronger ones per sampling, the appearance in this top NGRAMS_PER_DOC is one vote. The most voted NGRAMS_PER_PATH bigrams are considered to be related to the data path. We also experimentd with IDF weighting for the bigrams.

2.5 Ripper extraction

To obtain the right label for the training instance we do the following: for the exact-matched data paths, matched pieces of data will correspond to positive training classes, while unmatched pieces, negative ones. That is to say, if we know that

[11]Our bi-grams are computed after stop-words and punctuation is removed, therefore these examples can appear in texts like *"he is an **screenwriter**,**director**,... "* or *"she **has** an **screenwriter** award...*

(⟨`brother age`⟩ , 26) and that 26 appears in the text, we can conclude that the data of this particular person can be used as a positive training instance for the case (⟨`age`⟩ , 26). Similarly, if there is no match, the opposite is inferred.

For the U-test selected paths, the situation is more complex, as we only have clues about the importance of the data path as a whole. That is, while we know that a particular data path is relevant to our task (biography construction), we do not know with which values that particular data path is being verbalized. We needed to obtain more information from the sampling process to be able to identify cases in which we believe that the relevant data path has been verbalized, and that is why we turned to the n-gram distillation process described in the previous Section.

We use to `ripper`[12] (Cohen, 1996), a supervised rule learner categorization tool, to elucidate these types of relationships. A machine learning algorithm, `ripper`, takes as input a fixed-size fixed-typed feature-vector representing the relevant information for each instance, together with the target class for the instance (supervised learning). It produces as output decision rules that operate on feature vectors of the same type and size as the ones used in training and will chose the appropriate class, based on the training.

The ripper module takes the full data for a person, together with the selected labels and transforms this information as training material for a classification system. We trained a binary classifier (selected/unselected) for each individual piece of data. The main problem resolved in this module is how to represent using flat feature vectors a graph-based knowledge representation. Our solution to this problem is outlined below.

The problem of transforming a structured knowledge representation to a flat attribute-vector is a recurring one in machine learning We employed the following algorithm to propositionalize our input graphs (shown in Figure 4): First, we turned the graph into a tree, by fixing the node discussing the person being described as the root of the tree and successively building an arbitrary path to each node as the path to the root. Each of these trees could be traversed to obtain a flat vector. However, we wanted a sole vector to be able to represent all possible (available) trees, with a fixed coordinate in this vector to correspond across trees. We needed a means to record and fix the possible keys appearing at every node, together with the possible number of values each of these keys can take (in the case of multiple-valued keys, like `award` or `relative`). For this purpose, we engage in a process of finding a **unifier** tree for all the training trees. The process of building such a tree is summarized in Figure 7.

Once we have computed such unifier, a flat attribute-vector representation can be obtained by simulatenously traversing both the unifier and the tree at hand, producing default values in case of non-existent data in the target tree or the occurring value otherwise. Finally, we assign types to each coordinate of the output vector. We identify three types: numerical values (Ripper type `continuous`), strings belonging to a small subset of possible values (Ripper enumeration type) and string from an open set (Ripper type `bag`).

This flattening process generated a large number of features, e.g., if a person had a grandmother, then there will be a "grandmother" column for every person. This gets more complicated when list-valued values are taken into play. In our biographies case, an average-sized 100-triples biography spanned over 2,300 entries in the feature vector.

2.6 Other modules

Aside from the modules described above, there was a rule combination module; after the training data for each data path is generated, (cf. Figure 5(5)), the selected or unselected label will thus be chosen either via direct extraction from the exact match or by means of identification of distiled, relevant n-grams. After `ripper` is run, the obtained rules are our sought **content selection rules** (cf. Figure 5(5)).

3 Experiments

3.1 Corpus building

We obtained the semantic input from pages describing itemized factual information about persons: **fact-sheet** pages (Figure 8). Such pages are abundant for *celebrities* (singers, movie stars and other

[12] We chose `ripper` to use its set-valued attributes, a desirable feature for our problem setting.

famous people).

However, the level of detail of the fact-sheets was too coarse for our needs; we employed a combination of Information Extraction (IE) scripts to break the fact-sheet pages into smaller parts. A full-fledged natural language generation system expects a level of detail in its input that goes beyond the lay person ontological modeling found in the mined fact-sheets. To improve over that scenario, we performed a process we termed *Closed Information Extraction*. In this process, we build scripts without generalization in mind. It can be thought of as over-fitting a regular IE process.

Our current corpus contains semantic information for about 1,100 people, together with aligned biographies in three different sub-corpora, linked against 108, 578, and 205 biographies for each of these sub-corpora. The sub-corpora were mined from three different Web sites and contain different writing styles and text structure. Such an environment is ideal for learning and testing content planning related issues.

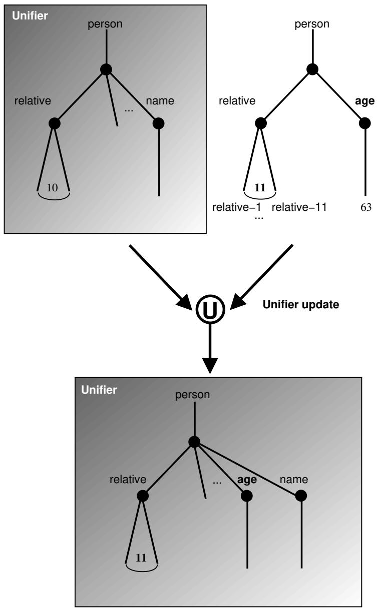

Figure 7: Computing the most general unifier for a set of structured inputs. A unifier in the process of being computed is updated with a new tree. The tree contains an unseen key (age) and more elements (11) in an already seen key (relative). Both elements get updated in the unifier.

3.1.1 Acquisition Process

The acquisition process consisted roughly of three steps: crawling, cleaning and linking. The crawling step involved downloading the actual pages. The cleaning step was the most time-consuming step and involved the *Closed Information Extraction* process mentioned in the introduction (and explained in more detail below). Finally, we had data about people and biographies about (hopefully the same) people. The last step took care of linking data with biography, when appropriate.

We crawled the fact-sheet pages from the *E! Online* Web site.[13] We could not find a central directory, so we proceeded to do a sequential scan over their people ID. The process involved several days of probing. To avoid overloading their server, we employed a random delay of 1 to 10 seconds between page fetches. While thousands of people were included in *E! Online* database, only a little more than a thousand contained fact-sheets with more than ten entries.

While *E! Online* contains also biographies, chances are the fact-sheets are based on the biographies or vice-versa. To improve the quality of the

[13] `http://www.eonline.com`

(a) Biography (b) Fact-sheet

Figure 8: Biography and fact-sheet page, from the Web.

aligned corpora, we mined different sites for the biographies. While several Web sites offer biographical information repositories, most of them focus in particular type of individuals. The celebrities domain is not fond of encyclopedia-style biographies, a person has to be famous for a long period of time before appearing in an encyclopedia and *celebrities* is a dynamic domain. We learned that lesson the hard way, after crawling a whole encyclopedia-style biographical repository and finding almost no links with our fact-sheet database. The sites we crawled that made up our corpus are: `biography.com`, `s9.com` and `imdb.com`. The `biography.com` repository was crawled by scanning the person id space. The `s9.com` site was crawled by probing on their "search" feature. The `imdb.com` Website was the easier to crawl, as they provide an index. Each of them has at least 20K different biographies. The `imdb.com` repository has in some cases more than one biography per person. The biographies differ in length, with an average of 450 words, 20 words and 250 words, for `biography.com`, `s9.com` and `imdb.com`, respectively. These differences in length imply differences in the selected information (hopefully also on the structure) that make them more interesting for learning content planning elements.

The crawled information can be used for research purposes only. We intend to use it to learn content planning elements, but it will not be presented to end users or publicly distributed. The pages were obtained by crawling. Special licensing arrangements would be needed for open distribution.

Regarding the cleaning step, it was the most time-consuming step, as already mentioned. This was no surprise, as data cleansing is normally considered among the most time consuming steps in data mining (Rahm and Do, 2000).

We cleaned the **fact-sheets** by means of a *Closed Information Extraction* process. The fact-sheets originally contained information in 14 categories, shown in Table 1. While some categories contained information that could be directly included in our knowledge representation (we use a representation similar to RDF (Lassila and Swick, 1999)), others contained heavily aggregated information (some difficult cases are shown in Figure 9).

To cope with these cases, we wrote a series of scripts, with patterns to capture different types of information. As usual, the patterns had errors, most frequently over-generating. We collected lists of the output yielded by these patterns (*gazetteers*) and manually cleaned them. By doing this, we made sure that the filler of `institution-of-study`

Person	Education	Agency
Factoids	Awards	Family
Birth Name	Occupations	Birth date
Quote	Birthplace	Significant Other(s)
Claim to Fame	Date of Death	

Table 1: Main categories in our semantic input.

award **1996:** London Film Critics' Circle: Best British Actor, *Trainspotting*; tied with Ian McKellen (*Richard III*) (E. McGregor)

education Yale University, New Haven, Connecticut; M.A., English Literature, 1987 Worked toward Ph.D.; did not complete thesis (D. Duchovny)

family **Mother:** Helen Barr, bookkeeper, cashier at Dee's Hamburger Drive-In; appeared with Roseanne on the Lifetime interview tribute special, *Like Mother, Like Daughter* (Roseanne)

significant other(s) **Husband:** Alan Hamel, producer, manager; met while both worked on *The Anniversary Game*; married 1977 (S. Somers)

Figure 9: Examples of the data that make up our frames.

will always be a true institution of study. During the process, we run into cases where the original wording made it almost impossible to write extraction patterns. We dealt with them in case by case basis and normally changed the wording by hand. We employed a total of 15 gazetteers (with a total of more than 30K entries), 12 of which where cleaned by hand (with a total of more than 2K entries). In essence, the process we followed is an over-fitted version of an information extraction process. We believe this *Closed Information Extraction* can be of help in the construction of aligned corpora of text and semantics.

The cleansing of the **biographies pages** was the usual task of extracting a section from a complex of HTML page, full of mark-up, advertisements, etc. This cleansing was accomplished by writing by hand a *hierarchical wrapper* (Muslea et al., 2001). We did not judge necessary to try to automatically learn such wrappers, as we were dealing with only three of them. This fact was another benefit of crawling whole sites instead of collecting assorted biographies by Web search. The cleaning process was concluded by expanding entities and normalizing character encodings.

Having a sizable set of semantic inputs and several sets of biographies as separated resources involved a real problem when it was the time to put them together. While we did not hesitate in spending hours of human labor for the construction of this corpus, aligning 1,100 fact-sheets against 20,000 biographies is truly unfeasible to be done by hand.

We thus needed to link the two resources, a step complicated by the fact that names tend to be paraphrased, and are not unique identifiers (e.g., there is a silent-movies era actor also named *Harrison Ford*). Moreover, linking a biography with the wrong knowledge base will considerably increase the overall noise for the machine learning machinery applied later on. To help us, we used techniques from *record linkage* in census statistical analysis (Fellegi and Sunter, 1969), a well studied discipline with more than 40 years of practice.

Fellegi and Sunter work on the task of classifying pairs in a product space $\mathbf{A} \times \mathbf{B}$, from two files A and B into $\mathbf{M}$, the set of true links and $\mathbf{U}$, the set of true non-links. To do that, the ratios of probabilities of the form:

$$R = \frac{Pr(\gamma \in \Gamma | M)}{Pr(\gamma \in \Gamma | U)}$$

are considered, where γ is an arbitrary agreement pattern in a comparison space Γ. Normally, Γ consists of eight patterns representing simple agreement or disagreement on *matching variables* (e.g., surname and age). The classical work of Fellegi and

Sunter defined an *optimal decision rule*, given by:

If $R > Upper$**,** then designate pair as a link.

If $Lower \leq R \leq Upper$**,** then designate pair as a possible link and hold it for human review.

If $R < Lower$**,** then designate the pair a non-link.

This decision rule is optimal in the sense that it minimizes the middle region over all decision rules on the same comparison space. The cutoff threshold $Upper$ and $Lower$ are determined by the error bounds and they provide methods to estimate them from the files **A** and **B** themselves. In our setting, we based our record linkage on the *Last Name, First Name,* and *Year of Birth* attributes.

3.1.2 Corpus Characteristics

This section summarizes major highlights of the constructed corpus. We briefly describe the knowledge representation, and report total figures of frames, relations, words, tokens and links.

We employ a type-based frame structure, with inheritance. Our XML-based encoding is roughly based on RDF, the encoding used in the Semantic Web. The information for each person is kept in a separate file, as a set of frames. Each frame has a unique name, a type (linked to an ontology) and a list of attribute-value pairs. Following RDF nomenclature, we count triples of the form (*frame name*, *attribute*, *value*). Attributes can be list-valued, can refer to other frames, or may contain atomic values (currently of types *symbol, string,* or *number*).

The final corpus contains 50,000 frames, with 106K frame-attribute-value triples, for the 1,100 people mentioned in each fact-sheet. The frames are linked through 43 different relations shown in Table 2. An example set of frames is shown in Figure 4.

The biographies side of the corpus can be measured in two different ways: unlinked and linked. The unlinked corpus contains all the downloaded biographies. It contains 2.8M, 488K, 2.5M tokens, for 21K, 25K, 15K biographies, from `biography.com`, `s9.com` and `imdb.com`, respectively. This unlinked resource is extremely useful for deriving language models of the domain. The linked resource, contains the biographies identified to be related to semantic input. Its size is 54K, 21K, 64K tokens for 108, 578 and 205 people, in the `biography.com`, `s9.com` and `imdb.com` Web-sites. There is an overlap of 170 double links (people data linked to two biographies) and seven triple links (people data linked to three biographies). `imdb.com` also contains some double biographies.

agency	father	province
aka	first	quote
award	full	reason
birth	givenname	relative
canned-text	last	significant-other
city	major	source
claimtofame	month	start
country	mother	subtitle
date	name	teaching-agent
day	occupation	text
death	older	title
degree	place	xtra
education	postmod	year
end	premod	younger
factoids		

Table 2: Relations in our biographical knowledge base.

3.2 Analysis of the experiments

We performed two full phases of training and testing. For development, we employed 102 biographies from `biography.com`, split into development training (91) and test (11), with a hand-tagged test set (average length of 450 words). For the final testing, we employed 205 new biographies from `imdb.com`, split into training (191) and test (14), with a hand-tagged the test set (average length of 250 words). As the length of the biographies were different, the content selection rules to be learned were different. We employed the development corpus throughout all the development process, where we debugged our code and fitted the various parameters described in the next section.

The annotation was done by one of the authors, by reading the biographies and checking which triples (in the RDF sense, (frame, slot, value)) were actually mentioned in the text (going back and forth to the biography as needed). Two cases required special attention. The first one was *aggregated information*, e.g., the text may say *"he received three Grammys"* while in the semantic input each award was itemized, together with the year it was received, the reason and the type (Best Song of the Year, etc.). In that case, only the **name** of award was selected, for each of the three awards. The second case was factual errors. For example, the biography may say the person was born in MA and raised in WA, but the fact-sheet may say he was born in WA. In those cases, the *intention* of the human writer was given priority and the place of birth was marked as selected, even though one of the two sources were wrong. The annotated data total 1,129 triples. From them, only 293 triples (or a 26%) were verbalized in the associated text and thus, considered selected. That implies that the "select all" tactic ("select all" is the only trivial content selection tactic, "select none" is of no practical value) will achieve an F-measure of 0.41 (26% prec. at 100% rec.).

3.2.1 Parameter fitting

Following the methods outlined in Section 2, we utilized the training part of the development corpus to mine Content Selection rules. We then used the development test to run different trials and fit the different parameters for the algorithm. Namely, we determined that filtering the bottom 1,000 TF*IDF weighted words from the text before building the n-gram model was important for the task (we compared against other filtering schemes and the use of lists of stop-words). The best parameters found and the fitting methodology are discussed below.

We then evaluated on the rest of the semantic input (998 people) aligned with one other textual corpus (`imdb.com`). As the average length-per-biography are different in each of the corpora we worked with (450 and 250, respectively), the content selection rules to be learned in each case were different (and thus, ensure us an interesting evaluation of the learning capabilities). In each case, we split the data into training and test sets, and hand-tagged the test set, following the same guidelines explained for the development corpus. The linkage step also required some work to be done. We were able to link 205 people in `imdb.com` and separated 14 of them as the test set.

We implemented the modules described in Section 2 as assorted Java classes and fitted various parameters using the development corpus. Namely, we fitted the following parameters:

Java Module `GenerateLogic`. This module consolidates the content selection rules logic generation; it is a top level module that generates the three logics marked as (A), (B) and (C) in Figure 5. Parameters:

- **CRISP_THRESHOLD.** Integer. This threshold is used to decide whether to use the exact-match derived logic or the class-based one. Employed in the **logic combination module**, to produce output (B).
- **USE_STOP_WORDS.** Boolean. Whether or not to use stop words when building language modules. Employed in the **language models construction module**, during (3).
- **STOP_WORDS_FILE.** String (filename). File containing the stop words (three where employed, the stop words from PubMed,[14] all the words but nouns and verbs from the lexicon of the Brill part-of-speech tagger (induced from the Penn TreeBank) and all the words but nouns, verbs and adjectives, from the same lexicon.
- **FILTERED_WORDS.** Integer. If no stop words file is employed (USE_STOP_WORDS is **false**), number of lowered scored TF*IDF values to discard as stop words (dynamic computation of stop words).
- **NGRAMS_PER_DOC.** Integer. Number of n-grams to pick per document in the sampling process. Employed in (3).
- **NGRAMS_PER_PATH.** Integer. Number of n-grams to pick per data path in the sampling process. Employed in (3).
- **RIPPER_OPTIONS.** String. Options passed to the Ripper executable. Employed in the rule induction module, used to obtain output (C).

[14] `http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html#Stopwords`

FEATURE_SELECTED_RIPPER. Boolean. Whether or not to reduce the number of features passed to Ripper by removing all data paths **not** selected by the class-based rules. Employed in the rule induction module, used to obtain output (C).

Java Module `ExactMatchRipperRules`. This module generate a set of ripper training data for each data path of interest. It is employed in the rule induction module, to obtain output (C).

LINES_PER_FILE. Integer. Number of lines in the Ripper training material that can be spanned by a single document-data pair.

ONLY_TOKENSTR. Boolean. Whether or not to generate the full training material or to restrict the training material to the string value of the data over which the decision is to be taken.

Java Module `CrossEntropyMatcher`. This module glues together the functionality of the clustering (2) and statistical selection (3) modules.

CLUSTER_THRESHOLD. Integer. Minimum size (in number of document-data pairs) allowed for each of the resulting clusters.

USE_DF_FILTERING. Boolean. Whether or not to use DF filtering to obtain dynamic stop word lists.

DISJUNCT_SETS. Boolean. Whether or not the cluster may overlap.

CLUSTERING_METHOD. Integer. Clustering method (distance-based, agglomerative with cut-off or agglomerative with catch-all.

TFIDF_SCORE_NGRAMS. Boolean. Whether or not employ TF*IDF scoring of n-grams (different than DF scoring).

The default values are shown in Table 3. After the first pass of parameter fitting, we picked the values shown in Table 4.

We found that choosing 500 n-grams per path in the distillation process produced optimal results and determined that the centroid-based one described in Section 2.2 achieved the best results. The best configuration is highlighted in Table 7.

Prec	Rec	F
0.406	0.731	0.522

Table 5: Exact match baseline for the parameter fitting

3.2.2 Stability analysis

To provide an idea of the statistical significance of the results, we performed two perturbation tests. These testes were a feasible alternative to cross-validation, prohibitive out of the expense of hand tagging. In these perturbations experiments, we trained in all data minus one instance (first experiment) or tested in all data minus one (second experiment). We performed both experiments on the ImDB corpus, the results are showed in Table 8.

3.2.3 Analysis of errors

The results are shown in Table 9. Several things can be noted in the table. The first is that `imdb.com` represents a harder set than our development set. That is to be expected, as `biography.com`'s biographies have a stable editorial line, while `imdb.com` biographies are submitted by Internet users. However, our methods offer comparable results on both sets. Nonetheless, the tables portray a clear result: the **class-based** rules are the ones that produce the best overall results. They have the highest F-measure of all approaches and they have high recall. In general, we want an approach that favors recall over precision in order to avoid losing any information that is necessary to include in the output. Overgeneration (low precision) can be corrected by later processes that further filter the data. Further processing over the output will need other types of information to finish the Content Selection process. The **class-based** rules filter-out about 50% of the available data, while maintaining the relevant data in the output set.

An example rule from the `ripper` approach can be seen in Figure 10. The rules themselves look interesting, but while they improve in precision, as was our goal, their lack of recall makes their current implementation unsuitable for use. We have identified a number of changes that we could make to

PARAMETER	VALUE
GenerateLogic.USE_STOP_WORDS	true
GenerateLogic.FILTERED_WORDS	1000
GenerateLogic.CRISP_THRESHOLD	7
GenerateLogic.NGRAMS_PER_DOC	1000
GenerateLogic.NGRAMS_PER_PATH	100
GenerateLogic.RIPPER_OPTIONS	
GenerateLogic.FEATURE_SELECTED_RIPPER	false
ExactMatchRipperRules.ONLY_TOKENSTR	false
ExactMatchRipperRules.LINES_PER_FILE	5
CrossEntropyMatcher.DISJUNCT_SETS	false
CrossEntropyMatcher.USE_DF_FILTERING	false
CrossEntropyMatcher.CLUSTER_THRESHOLD	6
CrossEntropyMatcher.CLUSTERING_METHOD	1
CrossEntropyMatcher.TFIDF_SCORE_NGRAMS	true

Table 3: Default values for our experimental setting.

PARAMETER	VALUE
GenerateLogic.USE_STOP_WORDS	false
GenerateLogic.FILTERED_WORDS	1000
GenerateLogic.CRISP_THRESHOLD	7
GenerateLogic.NGRAMS_PER_DOC	500
GenerateLogic.NGRAMS_PER_PATH	500
GenerateLogic.RIPPER_OPTIONS	-!ns -L17
GenerateLogic.FEATURE_SELECTED_RIPPER	false
ExactMatchRipperRules.ONLY_TOKENSTR	false
ExactMatchRipperRules.LINES_PER_FILE	5
CrossEntropyMatcher.DISJUNCT_SETS	false
CrossEntropyMatcher.USE_DF_FILTERING	yes
CrossEntropyMatcher.CLUSTER_THRESHOLD	9
CrossEntropyMatcher.CLUSTERING_METHOD	0
CrossEntropyMatcher.TFIDF_SCORE_NGRAMS	true

Table 4: Default values for the second pass in our experimental setting.

Case	Class-based			Ripper + exact match			Ripper + cross entropy		
	Prec	Rec	F	Prec	Rec	F	Prec	Rec	F
default	0.409	0.912	0.565	0.414	0.499	0.453	0.404	0.632	0.493
default+cluster_threshold=5	0.416	0.984	0.584	0.414	0.499	0.453	0.426	0.735	0.539
default+cluster_threshold=7	0.409	0.912	0.565	0.414	0.499	0.453	0.405	0.626	0.492
default+cluster_threshold=8	0.409	0.902	0.562	0.414	0.499	0.453	0.414	0.626	0.498
default+cluster_threshold=9	0.414	0.851	0.557	0.414	0.499	0.453	0.432	0.595	0.500
default+clustering_method=0	0.409	0.933	0.568	0.414	0.499	0.453	0.402	0.636	0.493
default+clustering_method=2	0.415	0.824	0.552	0.414	0.499	0.453	0.419	0.557	0.478
default+crisp_threshold=5	0.411	0.923	0.568	0.414	0.499	0.453	0.404	0.632	0.493
default+crisp_threshold=6	0.411	0.923	0.568	0.414	0.499	0.453	0.404	0.632	0.493
default+crisp_threshold=8	0.409	0.912	0.565	0.414	0.499	0.453	0.404	0.632	0.493
default+crisp_threshold=9	0.409	0.912	0.565	0.414	0.499	0.453	0.404	0.632	0.493
default+lines_per_file=1	0.409	0.912	0.565	0.427	0.445	0.436	0.416	0.588	0.487
default+lines_per_file=10	0.409	0.912	0.565	0.414	0.499	0.453	0.406	0.639	0.497
default+lines_per_file=20	0.409	0.912	0.565	0.414	0.499	0.453	0.406	0.639	0.497
default+lines_per_file=3	0.409	0.912	0.565	0.393	0.472	0.429	0.378	0.581	0.458
default+ngrams=1000,10	0.409	0.912	0.565	0.414	0.499	0.453	0.406	0.499	0.448
default+ngrams=1000,1000	0.409	0.912	0.565	0.414	0.499	0.453	0.416	0.680	0.516
default+ngrams=1000,500	0.409	0.912	0.565	0.414	0.499	0.453	0.416	0.680	0.516
default+ngrams=100,10	0.409	0.912	0.565	0.414	0.499	0.453	0.405	0.534	0.460
default+ngrams=100,100	0.409	0.912	0.565	0.414	0.499	0.453	0.416	0.667	0.512
default+ngrams=5000,1000	0.409	0.912	0.565	0.414	0.499	0.453	0.416	0.680	0.516
default+ngrams=500,10	0.409	0.912	0.565	0.414	0.499	0.453	0.406	0.499	0.448
default+ngrams=500,100	0.409	0.912	0.565	0.414	0.499	0.453	0.405	0.636	0.495
default+ngrams=500,500	0.409	0.912	0.565	0.414	0.499	0.453	0.416	0.680	0.516
default+random_sel_ngrams	0.408	0.912	0.563	0.414	0.499	0.453	0.405	0.615	0.488
default+ripper_L=0.7	0.409	0.912	0.565	0.417	0.506	0.458	0.417	0.660	0.510
default+ripper_L=1.1	0.415	0.964	0.579	0.430	0.520	0.471	0.428	0.691	0.528
default+ripper_L=1.3	0.409	0.912	0.565	0.417	0.496	0.453	0.412	0.646	0.503
default+ripper_L=1.7	0.409	0.912	0.565	0.407	0.489	0.444	0.405	0.646	0.498
default+ripper_S=0.1	0.415	0.964	0.579	0.414	0.513	0.458	0.414	0.680	0.514
default+ripper=negated	0.409	0.912	0.565	0.395	0.469	0.429	0.395	0.615	0.481
default+ripper=only_tokenstr	0.409	0.912	0.565	0.482	0.329	0.391	0.449	0.486	0.467
default+stopwords={JJ,NN,VB}	0.408	0.933	0.567	0.414	0.499	0.453	0.408	0.670	0.507
default+stopwords={NN,VB}	0.408	0.933	0.567	0.414	0.499	0.453	0.410	0.653	0.504
default+stopwords=filter_df(100)	0.411	0.933	0.570	0.414	0.499	0.453	0.401	0.629	0.489
default+stopwords=filter_df(1000)	0.409	0.912	0.565	0.414	0.499	0.453	0.400	0.609	0.482
default+stopwords=filter_df(200)	0.411	0.933	0.570	0.414	0.499	0.453	0.426	0.663	0.518
default+stopwords=filter_df(500)	0.411	0.933	0.570	0.414	0.499	0.453	0.426	0.639	0.511
default+stopwords=filter_tfidf(100)	0.409	0.912	0.565	0.414	0.499	0.453	0.412	0.632	0.499
default+stopwords=filter_tfidf(1000)	0.409	0.912	0.565	0.414	0.499	0.453	0.400	0.609	0.482
default+stopwords=filter_tfidf(200)	0.411	0.933	0.570	0.414	0.499	0.453	0.426	0.663	0.518
default+stopwords=filter_tfidf(500)	0.409	0.912	0.565	0.414	0.499	0.453	0.422	0.629	0.505

Table 6: First pass of parameter fitting experiment data

Case	**Class-based**			**Ripper + exact match**			**Ripper + cross entropy**		
	Prec	Rec	F	Prec	Rec	F	Prec	Rec	F
second-pass (default)	0.427	0.902	0.579	0.460	0.315	0.374	0.486	0.486	0.486
second-pass+cluster_threshold=5	0.412	0.933	0.571	0.460	0.315	0.374	0.452	0.516	0.482
second-pass+cluster_threshold=6	**0.429**	**0.912**	**0.583**	**0.460**	**0.315**	**0.374**	**0.486**	**0.486**	**0.486**
second-pass+stopwords=filter_tfidf(1000)	0.427	0.902	0.579	0.460	0.315	0.374	0.486	0.486	0.486
second-pass+ngrams=1000,500	0.427	0.902	0.579	0.460	0.315	0.374	0.486	0.486	0.486
second-pass+ripper=only_tokenstr	0.427	0.902	0.579	0.468	0.308	0.372	0.492	0.479	0.485
second-pass+stopwords={JJ,NN,VB}	0.419	0.946	0.58	0.460	0.315	0.374	0.463	0.53	0.494
second-pass+stopwords={NN,VB}	0.412	0.895	0.563	0.460	0.315	0.374	0.451	0.479	0.465

Table 7: Second pass of parameter fitting experiment data

Exact Match	
Prec.	0.048809462611884
Rec.	0.0960672736177549
F*	0.0647271915212909
Class-based	
Prec.	0.0530846379453527
Rec.	0.130033963732913
F*	0.0751554559156742
Ripper	
Prec.	0.0629943469814565
Rec.	0.0794001395087309
F*	0.0704790836954127

(a)

Exact Match	
Prec.	0.019683177509597
Rec.	0.00730446290692955
F*	0.0180179213735594
Class-based	
Prec.	0.0211509298728448
Rec.	0.030036746639124
F*	0.0248889538204664
Ripper	
Prec.	0.0172442088634342
Rec.	0.0590235642672372
F*	0.0365779309508538

(b)

Table 8: Perturbation analysis results. Standard deviation for training (a) and testing (b) perturbation.

```
SELECT ⟨award subtitle⟩
   IF ⟨occupation_1 TYPE⟩ = director AND
      ⟨education_2 place country⟩ = USA AND
      ⟨award_5 title⟩ ⊅ Festival
```

Figure 10: Example rule, from the `ripper` output. It says that the subtitle of the award (e.g., "Best Director", for an award with title "Oscar") should be selected if the person is a director who studied in the US and the award is not of Festival-type.

improve this process and discuss them at the end of the next section. Given the experimental nature of these results, we would not yet draw any conclusions about the ultimate benefit of the `ripper` approach.

With respect to the matching process, our "educated guess" of using the priors of each ambiguous match turned not to be a good idea. We plan to enhance this task in future work.

Our overall impression of the clustering process is that it did not work as good as it could . We expect more improvement on this line. We are currently looking at Cobweb (Fisher, 1987) as an alternative. However, it seems more important to incorporate more knowledge into the clustering process.

But the central item to address are the problems with `ripper`: the instance representation does not contain enough information to tell one instance from another, aside from the actual value of the instance. For example, if the name is "Steven," the data may be included or not taking into account whether the class is "c-brother".

Finally, the n-gram distillation process can be enhanced by means of a search process of the optimal number of n-grams to distillate. We did some preliminary experiments of this approach, where a binary search is performed over all n-grams, stopping when the two sets are discriminated on the sampling process described in Section 2.3.

3.3 Conclusions

We have presented a novel method for learning Content Selection rules, a task that is difficult to perform manually and must be repeated for each new domain. The experiments presented here use a resource of text and associated knowledge that we have produced from the Web. The size of the corpus and the methodology we have followed in its construction make it a major resource for learning in generation. Our methodology shows that data currently available on the Internet, for various domains, is readily useable for this purpose. Using our corpora, we have performed experimentation with three methods (exact matching, statistical selection and rule induction) to infer rules from **indirect** observations from the data.

Given the importance of content selection for the acceptance of generated text by the final user, it is clear that leaving out required information is an error that should be avoided. Thus, in evaluation, high recall is preferable to high precision. In that respect, our *class-based* statistically selected rules perform well. They achieve 94% recall in the best case, while filtering out half of the data in the input knowledge base. This significant reduction in data makes the task of developing further rules for content selection a more feasible task. It will aid the practitioner of NLG in the Content Selection task by reducing the set of data that will need to be examined manually (e.g., discussed with domains experts).

We have also presented a corpus of aligned text and semantics in a biographical description domain, stressing on the availability of the data on the Internet. We believe our experience is valuable for other researchers interested in assembling similar corpora in other domains. The corpus itself is a large corpus compared to other semantically rich corpus mentioned in the literature (Barzilay and Lee, 2002; Karamanis and Manurung, 2002). The quality of the information contained is good enough to support our planned series of experiments. More importantly, it is easy to expand with new data (we have started looking at People's magazine[15] repository of fact-sheet pages) and biographies.

We find the results for `ripper` disappointing and think more experimentation is needed before discounting this approach. It seems to us `ripper` may be overwhelmed by too many features. Or, this may be the best possible result without incorporating domain knowledge explicitly. We would like to investigate the impact of additional sources of knowledge. These alternatives are discussed below.

In order to improve the rule induction results, we may use *spreading activation* starting from the par-

[15] `http://people.aol.com`.

Experiment	development				imdb.com			
	Selected	Prec.	Rec.	F*	Selected	Prec.	Rec.	F*
baseline	530	0.40	0.72	0.51	727	0.35	0.68	0.46
class-based	550	0.41	0.94	0.58	891	0.36	0.88	0.51
content-selection	336	0.46	0.53	0.49	375	0.44	0.44	0.44
test set	293	1.0	1.0	1.0	369	1.0	1.0	1.0
select-all	1129	0.26	1.00	0.41	1584	0.23	1.00	0.37

Table 9: Experiment results

ticular frame to be considered for content selection and include the semantic information in the local context of the frame. For example, to content select ⟨`birth date year`⟩, only values from frames ⟨`birth date`⟩ and ⟨`birth`⟩ would be considered (e.g., ⟨`relative...`⟩ will be completely disregarded). Another improvement may come from more intertwining between the exact match and statistical selector techniques. Even if some data path appears to be copied verbatim most of the time, we can run our statistical selector for it and use held out data to decide which performs better.

Finally, we are interested in adding a domain paraphrasing dictionary to enrich the *exact matching* step. This could be obtained by running the semantic input through the lexical chooser of our biography generator, ProGenIE,[16] currently under construction.

[16] http://progenie.cs.columbia.edu/

References

Liane Acker and Bruce W. Porter. 1994. Extracting viewpoints from knowledge bases. In *Nat. Conf. on Artificial Intelligence*.

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *EMNLP-2002*, Philadelphia, PA.

William Cohen. 1996. Learning trees and rules with set-valued features. In *Proc. 14th AAAI*.

Richard Cox, Mick O'Donnell, and Jon Oberlander. 1999. Dynamic versus static hypermedia in museum education: an evaluation of ILEX, the intelligent labelling explorer. In *Proc. of AI-ED99*.

Pablo A. Duboue and Kathleen R. McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *2003 Conference on Empirical Methods for Natural Language Processing (EMNLP 2003)*, Sapporo, Japan, July.

Pablo A Duboue, Kathleen R McKeown, and Vasileios Hatzivassiloglou. 2003. ProGenIE: Biographical descriptions for intelligence analysis. In *Proceedings of the NSF/NIJ Symposium on Intelligence and Security Informatics*, volume 2665 of *Lecture Notes in Computer Science*, pages 343–345, Tucson, Arizona, June. Springer-Verlag.

Ivan P. Fellegi and Alan B. Sunter. 1969. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, December.

Douglas H. Fisher. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, New York, June. ISBN 0521585198.

Gabriel Illouz. 2000. *Typage de données textuelles et adaptation des traitements linguistiques, Application à l'annotation morpho-syntaxique*. Ph.D. thesis, Universit Paris-XI.

Nikiforos Karamanis and Hisar Maruli Manurung. 2002. Stochastic text structuring using the principle of continuity. In *Proceedings of the Second International Conference on Natural Language Generation (INLG-2002)*, Ramapo Mountains, NY.

S. Kim, H. Alani, W. Hall, P. Lewis, D. Millard, N. Shadbolt, and M. Weal. 2002. Artequakt: Generating tailored biographies with automatically annotated fragments from the web. In *Proc. of the Semantic Authoring, Annotation and Knowledge Markup Workshop in the 15th European Conf. on Artificial Intelligence*.

Ora Lassila and Ralph R. Swick. 1999. Resource Description Framework (RDF) model and syntax specification. http://www.w3.org/TR/REC-rdf-syntax, February. W3C Recommendation.

Kathleen Rose McKeown. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, Cambridge, England.

Ion Muslea, Steven Minton, and Craig A. Knoblock. 2001. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86.

Dragomir Radev and Kathleen R. McKeown. 1997. Building a generation knowledge source using internet-accessible newswire. In *Proc. of the 5th ANLP*.

E. Rahm and H. H. Do. 2000. Data cleaning: Problems and current approaches. *IEEE Bulletin of the Technical Committee on Data Engineering*, 23(4), December.

Barry Schiffman, Inderjeet Mani, and Kristian J. Concepcion. 2001. Producing biographical summaries: Combining linguistic knowledge with corpus statistics. In *Proc. of ACL-EACL 2001*.

Sidney Siegel and John Castellan Jr. 1988. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill, New York, 2nd edition.

Somayajulu G. Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. 2001. A two-stage model for content determination. In *ACL-EWNLG'2001*.

Ben Taskar, Eran Segal, and Daphne Koller. 2001. Probabilistic clustering in relational data. In *Seventeenth International Joint Conference on Artificial Intelligence*, pages 870–876, Seattle, Washington, August.

Elke Teich and John A. Bateman. 1994. Towards an application of text generation in an integrated publication system. In *Proc. of 7th IWNLG*.