ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands
T + 31 88 003 6136
E info@itea3.org
W www.itea3.org

ITEA 3 is a EUREKA strategic ICT cluster programme

D2.3.1 Overview and Comparison of Existing Generators

ModelWriter

Text & Model-Synchronized Document Engineering Platform

Work Package: WP2
Task: T2.3 – Overview and Comparison of Existing Generators

Edited by: Claire Gardent

Claire Gardent <claire.gardent@loria.fr> (CNRS)

Date: 30-Apr-2015
Version: 1.0.0

Document History

Version	Author(s)	Date	Remarks
1.0.0	Claire Gardent	30-Apr-2015	Initial Release

Table of Contents

1. Introduction

Given some input and a given communicative goal (e.g., describing an entity or summarising the input data), Natural Language Generation (NLG) aims to produce a text that satisfies this communicative goal given the provided input.

Contrary to Natural Language Understanding, (NLU) whose input is text, the input to NLG is not fixed and can in fact vary a great deal. It can be some numerical or graphical input provided by some application. For instance, the FoG system developed by CoGenTex Inc (Goldberg et al 1994) takes as input a numerical and a graphical depiction of the weather and generate textual weather report in English and French. The input can also be text. Thus the STOP system developed by the University of Aberdeen (Reiter et al. 2003) produces a personalised smoking cessation leaflet from a user filled questionnaire about smoking attitudes, beliefs and history. And it can be formal representations as exemplified in the STORYBOOK project where a fairy tale is generated from an abstract story plan (Callaway and Lester 2002).

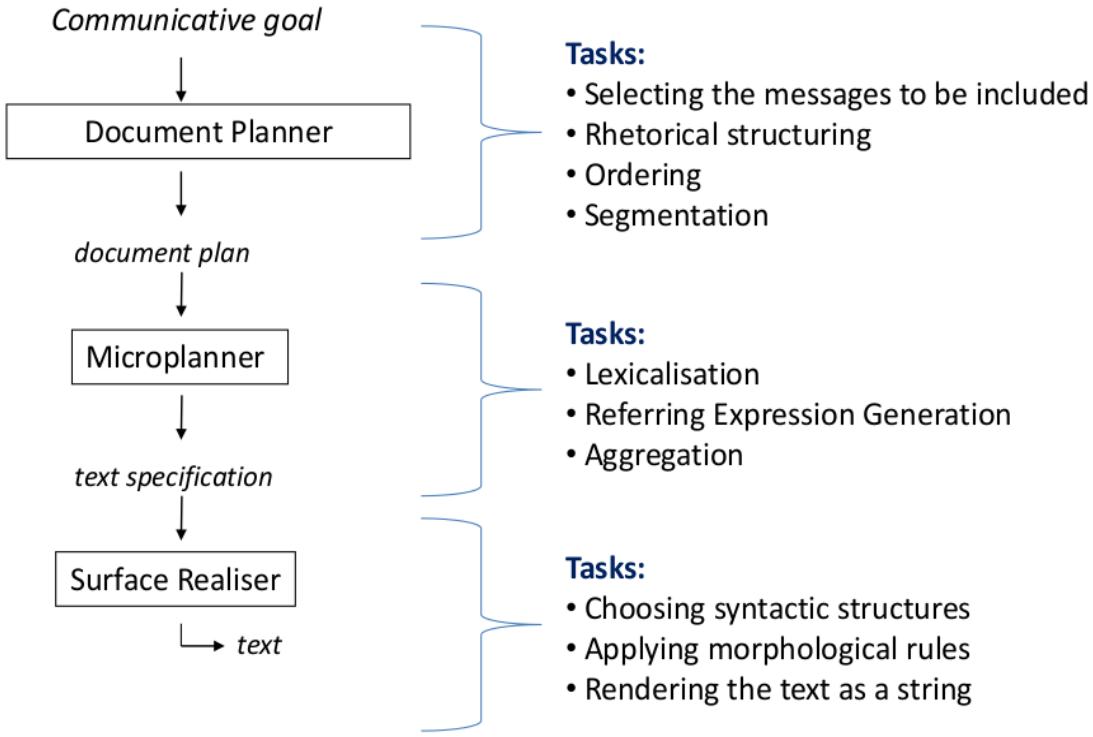
In the ModelWriter project, one main aim is to synchronise text and formal models. In particular, Work Package 2 (WP2) targets the definition, implementation and evaluation of a reversible process such that text can be automatically mapped to formal models (semantic parsing) and vice versa, models can be mapped to text (natural language generation). That is, in the ModelWriter project, NLG focuses on mapping formal representations to text. In this survey report, we therefore focus on on generation approaches which take formal representations as input.
While the input to NLG varies, the main research issues involved are constant across the board: content must be selected and structured; appropriate words and syntactic structures must be chosen; and referring expressions must be built which accurately supports the identification, by the reader, of the entities being talked about.

In this report, we survey different ways in which these main issues have been handled in the literature. Section 2 starts with a brief overview of NLG that summarises the key tasks that need to be performed by an NLG system. Although these tasks are known to be interdependent, in practice, they are often handled separately and integrated in a pipeline architecture. We therefore start by discussing the various types of symbolic, supervised and unsupervised or weakly supervised approaches which have been proposed for each of these key tasks (Sections 3 to 5). We then go on to survey those approaches which attempt to jointly capture all or several of the NLG subtasks (Section 6). Section 7 concludes with pointers for tools and techniques that are relevant for the ModelWriter project.

2. A brief Overview of NLG

While natural language understanding is mostly concerned with resolving ambiguity, natural language generation is mostly concerned with decision making i.e., with choice. Generating a text implies making choices about which content to verbalise (content selection), how to structure the selected content into a text plan (document planning), how to lexicalise a given input (lexicalisation), which syntactic structures to use (surface realisation), which content to group or to leave implicit (aggregation) and how to describe entities (referring expression generation). Figure 1 summarises the choices to be made and the terminology used. Document planning (also called Macro-Planning) focuses on selecting and structuring content before transforming it into linguistic representations while Microplanning addresses the various linguistic decisions that must be made once the overall document structure is determined. Surface realisation, also often included in the microplanning phase, consists in choosing the syntactic constructs to be used and in applying morphological constraints to ensure that the resulting text is morphologically and syntactically well-formed.

As mentioned in the introduction, the various choices required by the generation task are not independent of each other. However for practical purpose, the pipeline architecture illustrated in Figure 1 is often assumed wherein first document planning occurs, followed by microplanning, followed by surface realisation.



3. Document Planning

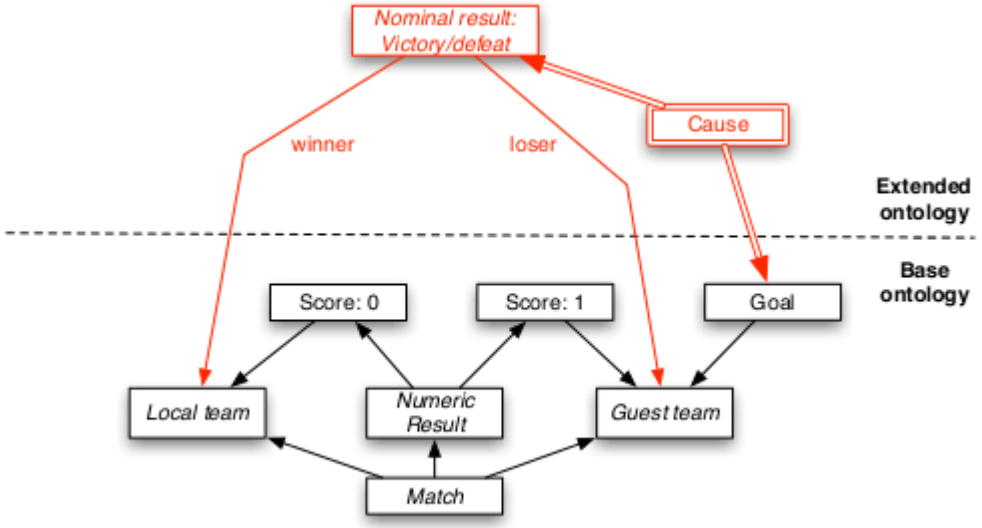
Document planning includes selecting the content to be generated from (*content selection*) and structuring this content into a document plan *i.e.,* segmenting it into basic units, ordering these units and determining the relation between each of these units. Existing approaches can be classified along different dimensions (type of input, bottom-up *vs* top-down approaches, pipeline *vs* joint architecture, etc). In what follows, we summarise existing work according to two main dimensions namely, content selection and content planning. For content selection, we review both top-down and bottom up approaches. For content planning, we survey different types of machine learning and statistical approaches.

3.1. Content Selection

3.1.1. Top-Down approaches

(Bouayad-Agha, Casamayor, and Wanner 2011) presents a top-down approach to content selection which proceeds in three steps. First, a subset of the knowledge base is identified (*content bounding*). Second, the main topics to be included in the content plan are selected (*main topics selection*). Third, discourse units are chosen (*fine grained content selection*).

The approach makes use of an extended ontology encoding the most frequently verbalised concepts and the semantic relations that implicitly hold between KB individuals. For instance, it includes the result (win or loose) of games which, although they can be derived from the KB, are not explicitly stated. This additional knowledge and the rules to infer it are obtained by manual analysis of a corpus of football match summaries.



The KB was automatically extracted from web pages about spanish football games and using a set of user-defined rules to populate the extended ontology. It contains 55894 instances.

The content bounding module selects KB individuals that are relevant to the game for which a text is generated using a set of hand-written rules.

Next the main topics module selects the most relevant concepts and relations using a simple user model and a set of heuristics. If a class is related to the user's team of interest, it is assigned a weight of 1, otherwise 0. Moreover this weight is multiplied by the class relevance measure which is set to 1 if the heuristic weight for selecting the instance outweighs the weight for not selecting it. Finally semantic relations are weighed with 1 if they link two nodes with a positive relevance weight. The weight of the instances is learned through data/text alignment and reflects the degree to which a data item is verbalised in game summaries. Boostexter (Schapire and Singer, 2000) is used to train a classifier which assigns relevance weight to instances.

Finally, the discourse unit determination module uses manually written templates to cover the types of propositions that are found in football summaries. For each node N that is the argument of a discourse relation, a set of paths defines its possible extensions i.e., the set of nodes (classes) that can be included in the discourse unit verbalising N.

The automatic alignment procedure is evaluated against 158 manually aligned summaries and yields an F score of 100%, 87% and 51% or red cards, goals and classification respectively. The evaluation of the content selection is done by comparing the content of the generated summaries with a gold standard. The test corpus comprises 36 matches each with 3 associated summaries from 3 different web sources. Precision and recall are obtained by comparing selected individuals/relations and against individuals/relations in the gold standard. Precision varies between 32.2% and 60.6% depending on the web source of the reference corpus.

3.1.2. Bottom-Up Open Planning

3.1.2.1. Content Selection as Natural Language Directed Inference

(Mellish and Sun 2005, Mellish and Pan 2008) address the problem of presenting parts of OWL DL ontologies in natural language. For instance, the axioms below might be verbalised as

> "A temporal region is a kind of region. An abstract region is also a kind of region but nothing is both a temporal region and an abstract region. One kind of temporal region is a time interval. A perdurant can happen at a time interval".

A10: $TemporalRegion \subseteq Region$
A2: $AbstractRegion \cap TemporalRegion = \bot$
A63: $TimeInterval \subseteq TemporalRegion$
A45: $Perdurant \subseteq \forall HappenAt.TimeInterval$
A51: $AbstractRegion \subseteq Region$

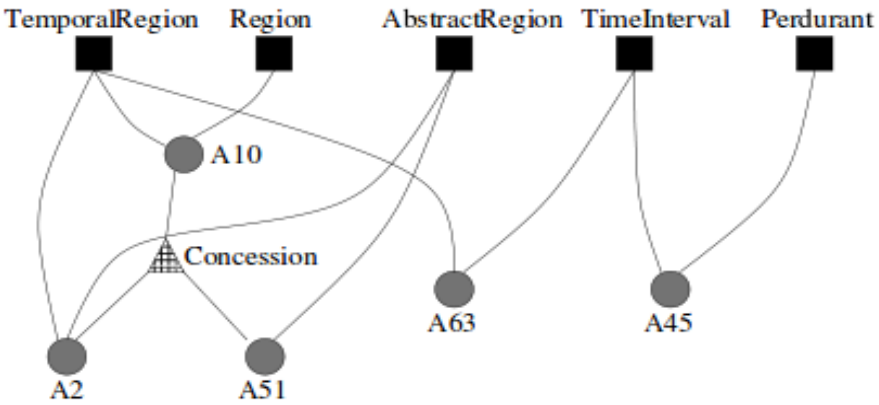
A basic approach to verbalise such set of axioms consists in associating axiom schemas with grammatically annotated templates. This might allow for instance, to map the axiom

$$Student \equiv Person \cap \exists Supervisor.Academic$$

to

> "A student is a person with at least one academic supervisor".

As illustrated below, axioms can be seen as forming a graph where each axiom is connected to the concepts it mentions and edge paths between axioms correspond to different possible transitions in a coherent text. Thus, given some input concept *X* to be described, a possible approach to content selection would consist in selecting axioms which are close (in terms of edges in the KB graph) to *X*, which are intrinsically interesting and which have not already been presented.



A10: $TemporalRegion \sqsubseteq Region$
A2: $AbstractRegion \sqcap TemporalRegion = \bot$
A63: $TimeInterval \sqsubseteq TemporalRegion$
A45: $Perdurant \sqsubseteq \forall HappenAt.TimeInterval$
A51: $AbstractRegion \sqsubseteq Region$

However this approach falls prey to the generation gap that is, it does not guarantee that the selected content supports the generation of a well-formed text. It may result in over-complex sentences (when an axiom is too complex to be expressed in a single sentence). It may yield repetitive text (when several axioms with identical structure are selected e.g., $Student \subseteq Person, Student \subseteq UnEmployed$ *and* $Student \subseteq \exists Supervisor.Academic$). It may place inappropriate emphasis on entities and it may prompt incorrect implicature (e.g., if the selected axiom is $Student \subseteq \exists Supervisor.Academic$and the KB also contains the axiom $Student \subseteq 1Supervisor.Academic$).

To overcome these limitations, (Mellish and Sun 2005, Mellish and Pan 2008) proposes a generate and test approach where a text is built incrementally by first enumerating a large set of possible texts and then choosing between them using a linguistically aware evaluation function. The set of possible texts is generated by selecting not only axioms present in the KB but also logical consequences of these axioms that are licenced by an inference mechanism dubbed "natural language directed inference". In essence this inference process expands the set of KB axioms with additional axioms which are sound, relevant, are based on individual axioms rather than several and lead to simple and coherent texts (e.g., through semantic aggregation).

The approach is evaluated for feasibility and scalability only (no human based evaluation of the selected content is carried out) and shown to be feasible for medium sized ontology (between 50 and 100 K bytes).

3.1.2.2. Content selection as a classification task

(Duboue and McKeown, 2003) view content selection as a classification task and learn selection rules from a database and its corresponding corpus. The corpus was mined from different web sites and the semantic data from fact-sheet pages. The goal is to learn which semantic data is actually expressed in the text and which is not. The approach uses clustering, cross entropy between language models and a decision tree algorithm to induce different types of content selection rules.

The approach proceeds in three steps. First, exact matching is used to identify data values which appear in the text. This creates a set of baseline *exact match content selection rules* describing data values that are systematically selected. Second, semantic classes are clustered depending on their values (e.g., the age class will be clustered into 2 groups, young and old) and a language model is learned for the texts corresponding to each cluster. Cross entropy between language models is then used to determine whether the class should be selected. If the cross entropy of the language model associated with a semantic cluster and of the language model associated with a set of randomly selected documents is greater than chance, then the class is selected. This second step yields a set of *class based content selection* rules which describe classes that are relevant for the domain at hand. Finally, an annotated corpus is automatically built where each data value is annotated as selected or not selected. If a data value matches an exact match content selection rule, it is marked as selected. N-grams matching values selected by a class-based content selection rule are also marked as selected. Using this annotated corpus, RIPPER is used to extract the content selection rules that will be used for NLG. For instance, a content selection rule might stipulate that the subtitle of the award should be selected if the person is a director who studied in the US and the award is not of Festival type.

The approach was tested on 11 document-data pairs where data was manually annotated for selection (triples expressed in the text were annotated as “selected”. The annotated data total 1129 triples of which 293 (26%) were verbalised in the associated text. The class based rules performed best (R:0.94, P:0.41, F1:0.58) whereas the content selection rules proved to have good precision but low recall (R:0.53, P:0.46,F1:0.49). Exact matching rules yielded an F1 of 0.51 (P:0.40,R:0.72).

3.1.2.3. Content Selection as an Optimisation Task

(Barzilay and Lapata 2005) present an optimization approach designed to collectively select data base entries (events) which are a priori important and are inter-related.
Content selection is viewed as an optimisation problem where the goal is to minimize label assignments which violate linking and selection constraints between events.

Each event (database entry) is assigned an individual preference score indicating whether it should be selected or omitted using a boosting algorithm (BoosTexter, Shapire and Singer 2000) which combines many simple moderately accurate categorisation rules into a single highly accurate rule. The individual preference scores is taken to be the confidence score output by BoosTexter.

Important links between event are induced using a generate-and-prune approach.
A pair of entity *(a,b)* is linked by *Li,j,k* if *a* is of type *Ei* (e.g., PASSING), *b* of type *Bj* (e.g., RUSHING) and they have the same value for attribute *k* (e.g., PLAYER 1). More generally links are created between two events if they share entities. A chi square test is then used to filter links in which entities have a similar distribution of label values while the weight of each link are computed using simulated annealing (i.e. minimizing an objective function which is defined as the error rate on the development set).

The approach is applied to a database containing football related information. The text corpus consists of 468 game summaries taken from the official site of the American National Football league (436580 words, avg length 46.8 sentences). The database is created from the tabulated information provided by the site. Labels (selected or ommitted) are assigned to DB entries by aligning DB entries and text using word overlap and marking entries for which a verbalisation was found as “selected” and others as “ommitted”. The overall dataset contained 105792 instances of which 15% (68 summaries) are reserved for testing and 1930 for development. The accuracy of the automatic labelling procedure was assessed against a gold corpus of 5 games (52 alignment pairs) yielding a precision of 94% and a recall of 90.4%.

The approach achieves an F-scores of 60.15% compared with 49.75% for a classifier (links scores set to 0) and 40.09% for a majority baseline (defaulting to the majority class for each event type).

3.1.2.4. Unsupervised Graph-Based Approach

Demir et al, 2010 describes an *incremental graph-based ranking algorithm* to iteratively determine which information is relevant to a given request while taking into account *discourse history*, the *a priori importance* of a fact, how strongly a fact is *related* to an already selected fact and whether a fact is *redundant* with respect to selected facts.

The approach uses a weighted undirected graph whose vertices are labelled with facts and where the weight of each edge represents how important it is to convey the facts related by the edge together. The graph also includes a priority vertex which is connected to all vertices with an edge whose weight encodes the apriori importance of a fact for the user.

The importance score of a vertex is computed using the weighted PageRank metric (Brin and Page, 1998). Content selection is done by iterating over the input graph, selecting a vertex with high a priori relevance and iteratively selecting facts which are relevant and non redundant. Vertex weights are adjusted at each step in the interaction to capture relevance and redundancy with respect to the selected facts. Vertices that are related to selected facts are given higher score while vertices labelled with facts that are less relevant or that have already been selected are assigned lower scores. To determine which facts are relevant, important or redundant, the data is manually annotated as essential/possible/not important (for vertices) and period/entity/contrast (for edges). A period edge expresses a relation between two propositions spanning the same time period, an entity edge indicates an entity overlap between 2 facts and contrast a contrastive relations. These relations are respectively assigned lowest, middle and highest score for relevance.

The approach is tested in a system describing information graphics and shown to perform better than a baseline where facts are selected based solely on their PageRank level. By taking into account which facts have already been selected and promoting facts that are related to selected ones, the approach leads to text with higher discourse cohesion than a system simply running PageRank to select the highly rated propositions.

3.2. Content Planning

In a language generation system, a content planner typically uses one or more plans to structure the content to be included in the output text and to determine the ordering between content elements.

3.2.1. Combinatorial Pattern Matching

(Duboue and McKeown 2001) presents a supervised approach to automatically learn plans for generation based on semantic types of the input clauses resulting in a top-down planner for selecting and ordering basic output elements. They manually annotate 24 transcripts of medical briefings with semantic tags and start by applying combinatorial pattern matching to the semantic sequences present in the annotated corpus. The resulting patterns are then refined using clustering and counting procedures are used to estimate order constraints between those clusters.

Given a set of sequences, a minimum window size and a support threshold *t,* combinatorial pattern discovery permits finding maximal (L,W) patterns with support threshold *t* and above.

The (L,W) pattern used is of the form $\Sigma(\Sigma|?) * \Sigma$where Σrepresents the semantic tag alphabet and matching sequences of length W should have at least L positions filled (i.e., they are non wildcards characters).

The support of a pattern is the number of sequences that contains at least one match for this pattern.

Pattern detection proceeds in three steps. First (Scanning) the n-grams of the semantic tags sequences are listed by increasing size. Second (Generalising), (L,W)-patterns are created for each n-gram. Only (L,W)-patterns with support greater than the fixed threshold are kept. The process (scanning and generalising) is repeated until no pattern with enough support is found. Third (filtering), less specific patterns are pruned. That is, if p1 is more specific than p2, if both have offset lists[1] of equal size, p2 is filtered out. This yields the lists of maximal patterns which are supported by the training data.

Among the patterns found, many are very similar. Agglomerative *clustering* is used with an approximate matching distance measure between patterns to group together patterns that are similar.

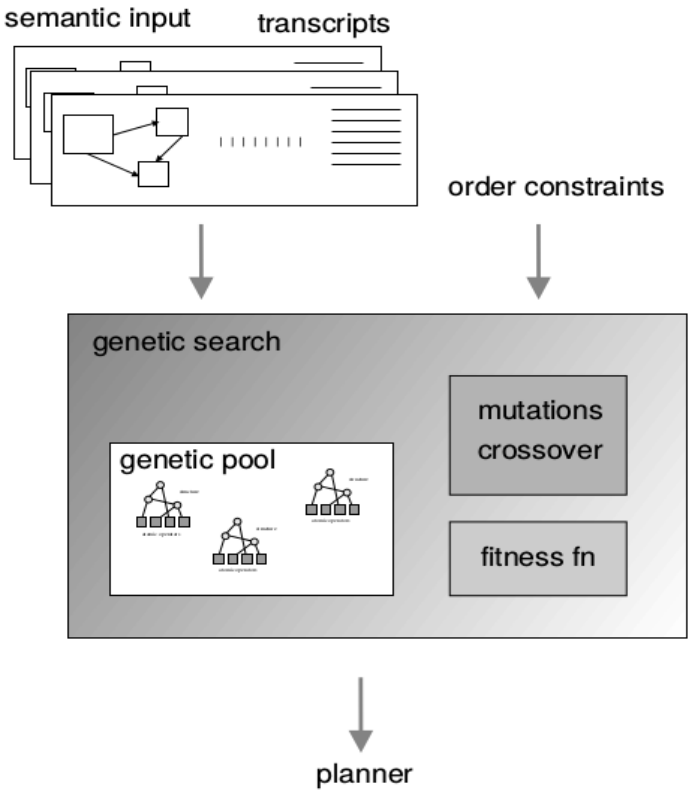
[1] The offset list records the matching locations of a pattern p in a list of sequences. It consists of sets of ordered pairs (seq,pos) where seq records the sequence number and pos the offset in that sequence where p matches.

Finally, ordering constraints between clusters are derived from frequency counts. In essence, frequent ones are kept while order constraints that are violated in any training sequence are rejected. The count *c* of *A* preceding *B* is normalised with the count *c1* of *A* preceding *x* and the count *c2* of *x* preceding *B* where *x* ranges over all the patterns that match before/after *A* or *B*. The arithmetic mean *((c/c1) + (c/c2))/2* is used as the final estimate for each constraint.

Given a training corpus of 24 transcripts, the system produced a set of 24 plan elements (semantic tag patterns) and 29 ordering constraints between these plan elements. For evaluation, these were compared using 3 fold cross validation with the original hand crafted plan that was constructed based on hand analysis of transcripts. For motif detection, L, W and support threshold were set to 2, 3 and 3 respectively. Pattern confidence i.e., the proportion of patterns that matches a sequence in the test set is 84.62%. Constraint confidence (the proportion of learned constraints for which there is at least one pattern from each cluster present) is 66.70%. Constraint accuracy (proportion of order constraints that are verified in all pairs of matching patterns in all the test set sequences) is 89.45%.

3.2.2. Evolutionary Algorithm

(Duboue and McKeown 2002) use evolution algorithms to learn a text planner determining the order of basic messages in medical briefings. The overall learning architecture is depicted in the diagram below. Given a set of 82 atomic messages that can occur in a briefing, the task is to learn a planner tree which will predict how to order these messages in an output text. The evolutionary algorithm explores the space of possible planning trees using mutation and cross-over operations to modify the current plan and two corpus driven fitness function to evaluate the fitness of each new tree.



Three mutation and one cross-over operations are used. The mutations include node insertion which picks an internal node at random and moves a subset of its children to a newly created subnode and node deletion which randomly picks an internal node different from the root and removes it by making its parent absorb its children. Both operators are order preserving. To allow for order variation, the shuffle mutation randomly picks an internal node and randomizes the order of its children. The cross over mutation is depicted in the Figure below.

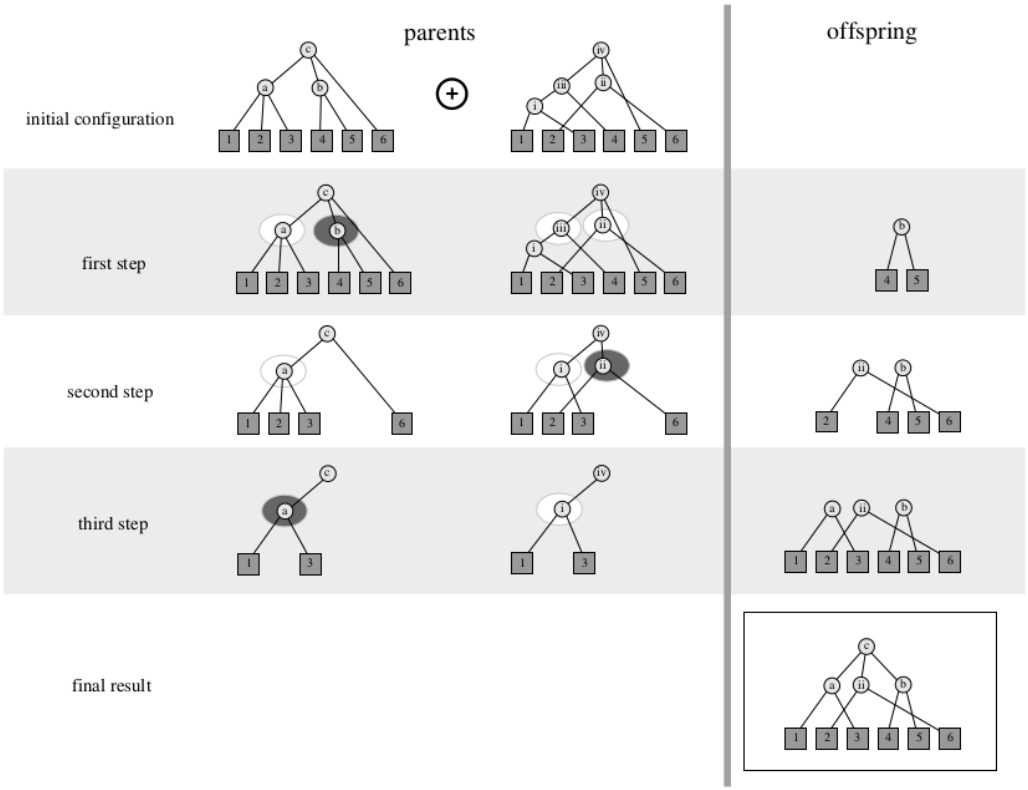


Figure 3: Cross over between chromosomes. Two trees are merged by picking subtrees of them (trees rooted at the dark circled nodes in 1st and 2nd step). Once a tree is moved to the offspring, all its leaves (and connecting nodes) are removed from both trees (note that node 6 is removed from both trees in step 3). The process continues until the parents are empty. This algorithm was chosen to maximize the amount of structure from the parents preserved in the offspring, its inspired on Bickel and Bickel (1987). Note that the first parent represents the ordering 123456, the second parent, 134265 while the offspring is 132645.

To assess the quality of a new text plan, first an approximate evaluation function Fc is used to determine whether order constraints over plan operators are met in the current plan. The order constraints are those acquired on the same domain in (Duboue and McKeown 2001). Once a tree has been evolved so that it conforms to all order constraints, the second fitness function *Fa* is used to assess how different the current text plan is from that generated by the existing MAGIC system developed for the same domain. Alignment between the two texts is used to determine how close the two texts are.

The search algorithm was initiated with a plan with one root node connected to a random ordering of the 82 basic messsages and executed by 20 generations. A comparison between the MAGIC and the best learned planner give a score of 1.16 where 0 would indicate a perfect match (the score estimates the number of distinct substructures output by the two planners). In comparison, the score of the initial random planner is 2.92.

3.2.3. A Statistical Approach

(Kan and McKeown 2002) learn from a corpus of semantically annotated bibliography entries (i.e., summaries) statistics which are used to determine both what predicates to include in a summary and how to order them.

A corpus of 2000 summaries is semantically annotated as follows. 5% (100) entries are manually annotated with a tagset of 24 predicates (e.g., purpose, audience, contributor, author). The Ripper decision tree learner is used to induce a decision tree that can automatically label a new corpus with predicates. Semantic annotation applies to node of the sentences dependency tree. The features used represent the predicate's set of words and relative and absolute position in the summary; local context information (i.e., preceding and following predicates); and genericity (how uniform the language is for particular predicates across instances). When testing using 5-fold cross validation, the resulting accuracy is 66%.

The semantically annotated corpus is the basis for learning the rule base for content planning. Content selection is done by a randomized algorithm which selects n (n is the user defined desired summary length) predicates biased for the percentages acquired from the training corpus. The selected predicates are then ordered using either the harmonic or quadratic penalised version of the algorithm. These are computed as follows.

n-gram statistics on pairs of adjacent predicates are recorded together with the order in which they occur. A precedence relationship of distance one is given a full strength score but a distance n relationship is given 1/n unit score in the harmonic and $1/2^n$ in the quadratic. Each pair of predicates accumulates these weights as instances are found in the training corpus and a randomised hill-climbing algorithm is used to find a maximally compliant ordering.

3.2.4. A Classification Approach

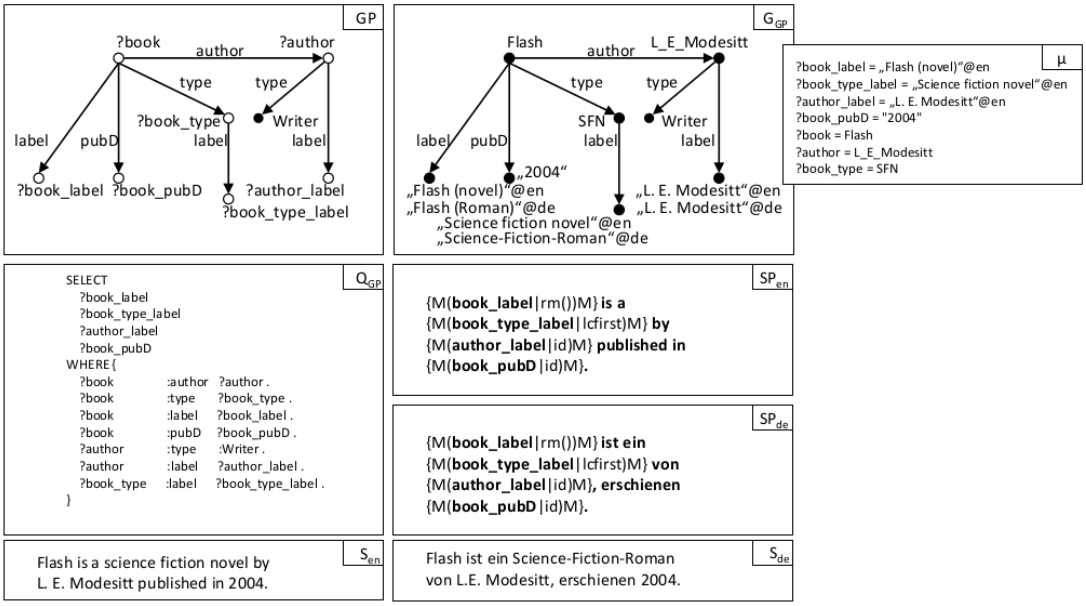
(Dimitromanolaki and Androutsopoulos 2003) decompose fact ordering into a cascade of multi-class classification problems where each classifier selects the fact to be placed at the corresponding position. The input to each classifier is a vector of binary feature indicating which fact classes are in the input and which have already been selected. The output is the class of the fact to be placed at the position being processed. The authors experiment with both an instance-based and a decision tree algorithm. They compare their results with two baselines, a majority baseline which assigns to position n the fact class that was most frequent at position n in the training data; and a predefined fixed order determined in collaboration with a domain expert. The accuracy (percentage of correct selections at each position) of the classifiers outperforms both baseline. The approach is trained and tested (using 10 fold cross validation and stratification) on a set of 880 combinations of 6 facts each which were manually ordered according to a domain expert recommendation.

4. Microplanning

4.1. Lexicalisation

4.1.1. Verbalisation Templates for RDF Subgraphs

(Ell and Harth 2014) present a language independent method for extracting RDF verbalisation templates from a parallel corpus of text and RDF data. A template consists of a graph pattern that can be used to query a RDF graph and of a sentence pattern which is a slotted sentence that will be completed by inserting in the slots the results of the query.



The approach consists of six steps.

1. For each entity in the RDF graph, sentences that contain mentions of that entity are extracted. Mentions are labels of the entity (as specified in DBPedia) modulo minor string modifications to account for morpho-syntaxic or typographical variations.
2. Each of those sentences is associated with a set of identified entities, a subgraph and a set of observation. An observations is a 7 tuples of the form (e,p,o,l,r,o',m) where e,p,o is an RDF triple, l and r are the strings matched to the left and the right of the mention of o, o' is the matched mention and m is the modifier (modification) applied for matching o and o'. An identifed entity is an entity e such that the sentence contains a (possibly modified) mention of e. The sentence subgraph consists of all triples that contain an identified entity.
3. Sentence and graph abstraction. Identified literals in the graph and in the sentence are replaced by shared variables. Different sentences may share the same graph pattern.
4. Grouping. Given a set of (sp,gp) tuples, each sentence pattern sp is grouped with the set of graph patterns found for it.
5. Frequent maximal subgraph pattern extraction. The maximal frequent subgraphs of a set of subgraph patterns are extracted.
6. Template creation. For each (sp,gp) tuple, the frequent maximal subgraphs of gp are extracted. Subgraphs that are not safe, not connected or which yield no results are filtered out.

The approach is tested on Wikipedia and DBPedia in English and in German. For 3 587 146 and 613 027 entities respectively, 3 434 108 (resp. 530 766) templates were extracted where at least 2 entites were identified . Evaluation includes coverage (numbfer of subgraphs verbalised by a

graph pattern), accuracy (proportion of triples expressed in the sentence and vice versa degree to which the sentence content is expressed by the sentence), syntactic correctness (degree to which a verbalisation is correct) and understandibility (clarity of verbalisation).

4.1.2. Verbalising RDF data using Ontology Lexica

(Cimiano et al 2013) use a manually constructed lexicon to verbalise ontology concepts. The lexicon conforms with the LEMON, a lexicon model for ontologies (McCrae et al 2011) and describes the link between a concept, its possible verbalisations and conditions on the use of these verbalisation. For instance, the following lexical entry lists two possible verbalisations for the ontology concept “schneiden” namely the infinitive form “schneiden” and the past participle form “geschnitten”.

```
:schneiden a lemon:LexicalEntry ;
 lexinfo:partOfSpeech lexinfo:verb ;

 lemon:canonicalForm [
 lemon:writtenRep "schneiden"@de ;
  lexinfo:tense lexinfo:present ;
  lexinfo:mood  lexinfo:infinitive
 ];
 lemon:otherForm [
  lemon:writtenRep "geschnitten"@de ;
  lexinfo:verbFormMood
                lexinfo:participle ;
  lexinfo:aspect lexinfo:perfective
 ];

 lemon:sense
 [ lemon:reference action:schneiden ].
```

Similarly, the entry below shows a mlexical entry for “tranchieren” (to carve) which refers to the same “schneiden” concept but is restricted to cases where the ingredient if of type “meat”. The condition is modelled as a logical condition than can be issued as a query on the knowledge base.

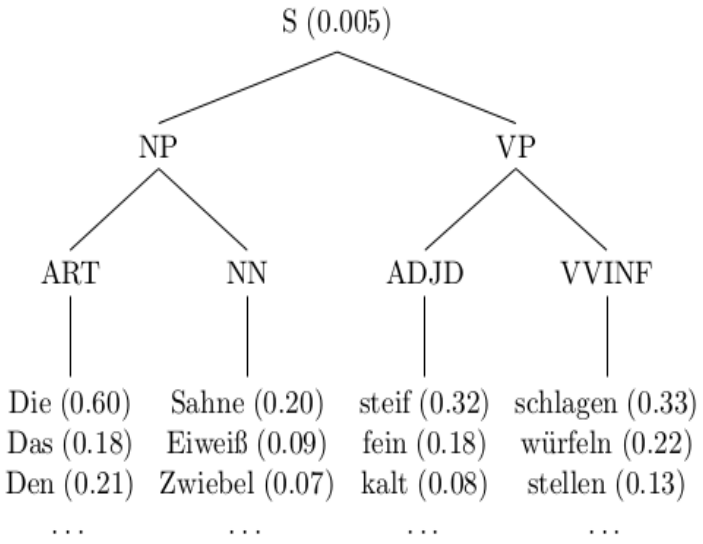
```
:tranchieren a lemon:LexicalEntry ;
 lexinfo:partOfSpeech lexinfo:verb ;

 lemon:canonicalForm [
  lemon:writtenRep "tranchieren"@de ]

 lemon:sense
 [ lemon:reference action:schneiden;
   lemon:condition [ lemon:value
    "exists ?x :
     :hasIngredient(?x,?y),
     :Step(?x),
     ingredient:Fleisch(?y)" ];
   lemon:context
     isocat:technicalRegister ] .
```

To verbalise a given concept, the lexicon is consulted and only these entries whose conditions are satisfied are selected. In addition corpus statistics are used to determine the terms or term combinations with higher frequency in the domain corpus.

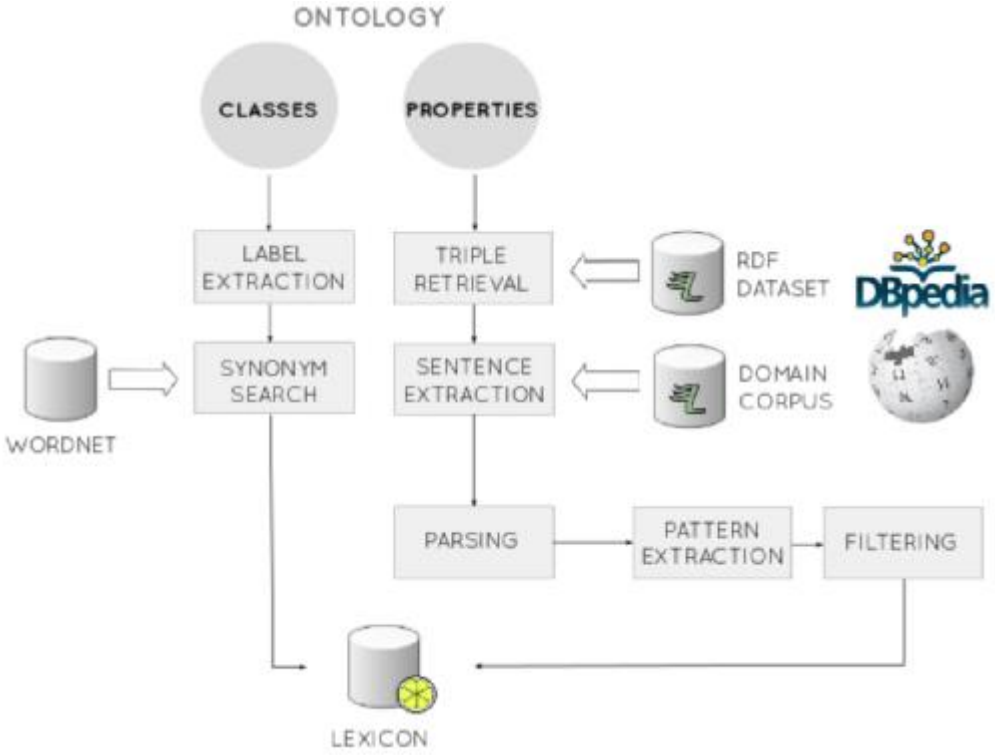
During surface realisation, selected lexical entries are combined using sentence templates induced from a domain specific corpus (e.g., cooking recipe). These templates are acquired by first parsing the corpus sentences and annotating them with the ontology concepts. 20 000 templates are extracted whose leaves are labelled with the list of all terms occurring at that position in the parse tree together with the corresponding senses (KB concepts) and the frequency of the terms.



At generation time, the input concepts are used to select appropriate lexical entries and the task of the surface realiser is to find an appropriate syntactic tree to realise the input concepts. The selected tree is a tree that maximises a score taking into account the normalised probability of the syntax tree (induced from frequency counts on domain corpus), a comparison of the part of speech tag, synonyms and lexical senses of each selected lexicalisation with those of the terms in the tree, the node distances of related words inside each tree and an n-gram score for each resulting sentences.

To automate the production of the lexicon, (Walter et al.) present a semi-automatic approach that exploits a corpus to find occurrences in which a given property is expressed, and generalises over these occurrences by extracting dependency paths that can be used as a basis to create lemon lexicon entries. The approach is evaluated with respect to DBpedia as dataset and Wikipedia as corresponding corpus.

The approach is summarised in the figure below.

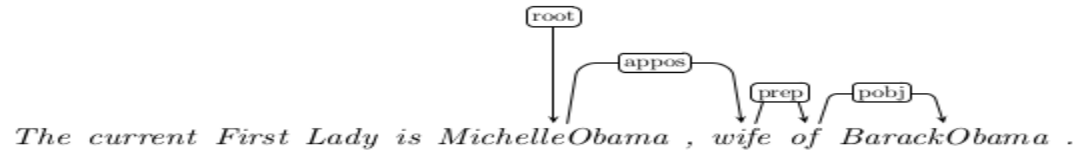


For each property to be lexicalised, all triples from the knowledge base containing this property are retrieved. The labels of the subject and object entities of these triples are then used to search the corpus for sentences containing both these entities. Patterns are then extracted from the dependency paths of these sentences and used to construct lexical entries.

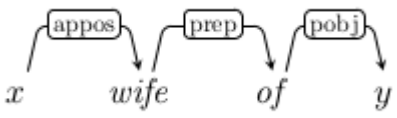
For instance, given the RDF triple

$$(\texttt{resource:Barack\_Obama}, \textit{dbpedia:spouse}, \texttt{resource:Michelle\_Obama})$$

The following sentence might be extracted and parsed.



From this parse, the following pattern will be extracted:



and the following lexical entry will be built:

```
wife a lemon:LexicalEntry ;
 lexinfo:partOfSpeech lexinfo:noun ;
 lemon:canonicalForm [ lemon:writtenRep "wife"@en ] ;
 lemon:synBehavior [ rdf:type lexinfo:NounPPFrame ;
                     lexinfo:copulativeArg       :x_appos ;
                     lexinfo:prepositionalObject :y_pobj ] ;
 lemon:sense [ lemon:reference
               <http://dbpedia.org/ontology/spouse>;
               lemon:subjOfProp :x_appos ;
               lemon:objOfProp  :y_pobj ] .

y_pobj lemon:marker [ lemon:canonicalForm
                    [ lemon:writtenRep "of"@en ] ] .
```

Patterns are generated by abstracting over the entities occuring in the parse, removing determiners and removing patterns with length less than 3 or more than 6. Also if the entities are related to another token by the nn relation (ie are modifiers) , the pattern is not kept. For each property, only those pattern whose relative frequency is above a given threshold are kept.

4.2. Generating Referring Expressions

A given target object can be referred to in many ways. Different forms may be used e.g., a pronoun (he), a proper name (John), a definite description (the man) or a demonstrative (this man). This choice can often be modelled by simple heuristics (e.g., using a proper name of a definite description for a first mention and a pronoun for subsequent mentions). For descriptions, additional choices concern the content of the description (which attributes to include?) and its form (how to verbalise each attribute). We focus here on the content selection problem.

4.2.1. Full Brevity, Greedy and Incremental Search

(Dale 1989, Dale and Reiter 1993) present three base algorithms that have been widely used for generating entity descriptions.

Given an input consisting of a target entity, a set of other objects (called “distractors”) and a set of attributes, the output is a set of attributes which uniquely identifies the target entity i.e., which eliminates all distractors. For instance, given the objects and attributes shown below, a distinguishing output description for $d_1$could be either {<type,man>,<clothing,wearing suit>} or {<type,man>,<position,left>} which could be realised as “the man wearing a suit” or “the man to the left”.

Object	type	clothing	position
d_1	man	wearing suit	left
d_2	woman	wearing t-shirt	middle
d_3	man	wearing t-shirt	right

The first, full brevity, algorithm searches for a minimal set of attributes that uniquely identifies the target entity. To find a minimal description the algorithm first checks whether there is a single property of the target that rules out all distractors i.e., that is false of all distractors. If that fails, all possibles combination of 2 properties are used, etc. The process repeats until either all distractors have been ruled out or all properties of the target have been tried.

One main drawback with this algorithm is that it is computationally expensive (NP hard). Moreover humans sometimes use non minimal descriptions. In contrast, the greedy algorithm incrementally selects that property which eliminates the highest number of distractors. This does not garantee a minimal description but is much more efficient (because not all combinations of properties are explored).

Finally, the incremental algorithm incrementally selects properties based on a predefined ordering e.g., by selecting the gender of a person before the color of her eyes.

As extensively discussed by (Krahmer and Van Deemter 2012) these basic algorithms have been extended and modified in multiple ways e.g., to take into account not only unary properties but arbitrary n-ary relations; to generate plural and vague descriptions; to recast the problem in terms of existing framework such as labelled directed graphs and description logics; and to evaluate or learn models and algorithm on empirical data.

4.2.2. A Corpus Based Investigation of Architectures

(Zarrieß and Kuhn 2013) consider referring expressions, syntax and word order and explore how different architectural setups account for their interactions. Using a corpus annotated with deep syntax and discourse referents, they develop a statistical approach which can map a deep syntax tree and a set of referents to a sentence. The approach combines a syntax generator mapping a deep to a shallow dependency tree, a referring expression generator and a linearizer. They combine these three modules in different ways and examine how these different combination modes impact the generated text.

4.2.3. Combining a Sentence Planner and a Maximum Entropy Model for Referring Efficiency

(Garoufi and Koller 2012) generates maximally useful referring expressions by combining a maximum entropy model for referential success trained on the GIVE-2 corpus -and using the model weights as costs in a metric sentence planner.

The GIVE-2 corpus is a corpus of instructions given in virtual environment (Gargett et al. 2010) to guide the user in a virtual world. The expressions referring to buttons (the objects manipulated by the user) are annotated with the type of the attributes they contain. Six main attributes types are considered.

RE attribute type	%
Absolute property (color; e.g. "red")	79.83
Taxonomic property (type; e.g. "button")	59.80
Viewer-centered (e.g. "on the right", "the left one")	19.33
Micro-level landmark intrinsic (e.g. "by the chair")	17.37
Macro-level landmark intrinsic (e.g. "next to the doorway")	8.54
Distractor intrinsic (e.g. "next to the yellow button")	7.00

To model the connection between the current 3D context and referring success, the following ten context features are used.

Object relations	
RoomSameTypeDisNum	the number of distractors of the same type as the referent in the room
MicroLandmarkInRoom	whether there are any micro-level (i.e. movable) landmarks in the room
MacroLandmarkNearby	whether there are any macro-level (i.e. immovable) landmarks near the referent
Spatio-visual	
Distance	the Euclidean distance (in GIVE space units) between the IF and the referent
Angle	the angle (in radians) between the center of the IF's field of view and the referent
Referent's distinctiveness	
ColorUnique	whether the referent's color is unique (i.e. not shared by other objects) in the world
LandmarkTypeUnique	whether a landmark with unique type in the world exists in the referent's room
Interaction history	
Round	the number of times the referent has been target of manipulation in a whole session
ReferenceAttempt	the number of times the referent has been referred to in the same round
SeenDeltaTime	the time elapsed (in seconds) since the referent was last seen by the IF

A maximum entropy model is then learned to estimate the successfulness of any RE (referring expression) in any context.

First, referring expressions in the training corpus are split into a class of high sucessfulness and one of low successfulness as follows:

$$succ^*(r) = 0 \; if \; succ(r) \leq S, 1 \; otherwise$$

where *S* is the median of all values that *succ(r)* takes and the successfulness $succ(r)$ of a referring expression *r* is defined as:

$$succ(r) = 0 \; if \; r \; was \; not \; correctly \; resolved, \frac{\Delta S}{\Delta T} \; otherwise.$$

$\Delta s$is the distance in the GIVE world between the target referent and the hearer's location at the time whey they are presented with the referring expression *r*. ΔT is the time elapsed between the presentation of the RE and the manipulation of the referent.

The model is trained using logistic regression to learn the conditional probability of an RE r issued in a scene s being successful given a joint representation of attributes and context:

$$P(succ^*(r) = 1 \mid \{\phi_{ij}(r,s)\}_{ij})$$

with $\phi_{ij}(r,s) = c_i(s).a_j(r)$where $a_j(r) = 1$if *r* contains an attribute of type a_j and $c_i(s)$takes the value of the feature c_i on the scene *s.*

The weight $v_j(s)$ of each attribute j in a given scene is then used to determine the cost of the planning operators used to build the natural language verbalisation of the set of selected attributes. As a result, a sentence plan has minimal cost among all correct plans just in case the referring expressions it contains has maximal successfulness probability.

5. Surface Realisation

5.1. Inverse Parsing

Given a grammar developed for parsing, surface realisation as inverse parsing consists in using this grammar to generate, rather than to parse, sentences. Inverse parsing approaches are typically developed for grammars which describe both the syntax and the semantics of natural language. They have been proposed for Lexical Functional Grammars, Tree Adjoining Grammars, Combinatory Categorial Grammar and Head Driven Phrase Structure Grammar.

(Bangalore and Rambow 2000) describes an approach composed of three modules: a stochastic tree chooser, a grammar-based unraveler and a stochastic Linear Precedence chooser. The approach takes as input an ordered dependency tree whose nodes are labeled with lexemes. The tree chooser (aka supertagger) then uses a stochastic tree model to assign TAG (Tree Adjoining Grammar) trees to the nodes of the input dependency tree. A node n is assignet a TAG tree t so that the probability of the subtree composed of n with supertag (ie. TAG tree) t and all of its daughter tags is maximal. The unraveller uses the TAG grammar to produce a lattice of all possible linearizations that are compatible with the supertagged tree (i.e., the input tree whose nodes are labelled with lexemes and TAG trees) and the grammar (linear order information is deduced from the TAG trees). Finally, the LP chooser chooses the most likely traversal of this lattice given a language model.

Another approach to inverse parsing with TAG is presented in (Narayan and Gardent 2012) who introduce a highly optimised surface realisation algorithm for TAG which exploits the structure of the input dependency tree to filter applicable rules and implement a beam search filtering the n best intermediate solutions using a language model. The approach is tested on the benchmark made available by the Surface Realisation Shared Task namely a set of unordered lemmatised dependency trees derived from the Penn Treebank.

(Carroll and Oepen 2005) present an inverse parsing approach for HPSG (Head Driven Phrase Structure Grammar) which combines symbolic optimisation techniques (tabulation, subsumption based local ambiguity factoring) with a conditional, discriminative model for ranking the output sentences. Similarly, (White et al 2007) use a language model, a beam search and a ranking model to prune the search space when generating with a Combinatory Categorial Grammar (CCG). Finally, for Lexical Functional Grammar (LFG), (Cahill and van Genabith 2006) propose a PCFG-based approach based on an LFG which was automatically extracted from a treebank.

5.2. A Maximum Entropy Framework for Surface Realisation

(Ratnaparkhi 2000) presents a surface realiser where the mapping between input semantic representations and natural language strings is learned from a corpus of “generation templates” i.e., sentences in which values of interest (semantic values) have been replaced with their corresponding attributes. For instance, given the set of semantic representation in (1a), the sentence in (1b) would yield the generation template in (1c).

1a. {$city-fr = New Yok City, $city-to = Seattle, $time-dep = 6am, $date-dep = Wednesday}
1b. A flight to Seattle that departs from New York City at 6am on Wednesday.
1c. A flight to $city-to that departs from $city-fr at $time-dep on $date-dep

Using this corpus of generation template and associated semantic representations, the authors learn three generation models. The first model (NLG1) chooses the most frequent template in the training data that corresponds to a given set of attributes. It fails if this set of attributes has not been seen in the training data. NLG1 is the baseline.

The second model (NLG2) is a conditional distribution model over the vocabulary V of words seen in the training data. The model is a maximum entropy model of the form:

$$p(w_i|w_{i-1}, w_{i-2}, attr_i) = \frac{\prod_{j=1}^{k} \alpha_j^{f_j(w_i, w_{i-1}, w_{i-2}, attr_i)}}{Z(w_{i-1}, w_{i-2}, attr_i)}$$

where $w_i$ranges over *V* and $\{w_{i-1}, w_{i-2}, attr_i\}$ is the history where $w_i$denotes the i-th word in the generated phrase and $attr_i$ denotes the attributes (in the input meaning representation) that remain to be generated at position *i* in the phrase. $f_i$are features and capture information in the history that is useful fo estimating $p(w_i|w_{i-1}, w_{i-2}, attr_i)$. The feature weights $\alpha_j$are obtained using the Improved Iterative Scaling algorithm and are set to maximise the likelihood of the training data. At generation time, the probability of the sequence of words $w_1 \dots w_n$ given the semantic representation A (attribute set) is:

$$p(w_1 \dots w_n \mid A) = \prod_{i=1}^{n} p(w_i|w_{i-1}, w_{i-2}, attr_i)$$

The features used are the following:

Description	Feature $f(w_i, w_{i-1}, w_{i-2}, attr_i) = \dots$
No Attributes remaining	1 if w_i =? and $attr_i = \{\}$, 0 otherwise
Word bi-gram with attribute	1 if w_i =? and w_{i-1} =? and ? $\in attr_i$, 0 otherwise
Word tri-gram with attribute	1 if w_i =? and $w_{i-1}w_{i-2}$ =?? and ? $\in attr_i$, 0 otherwise

Sentences are incrementally generated by choosing at each step the word sequence with highest probability given the already generated sentence and the semantic representation (attributes) remaining to be verbalised.

NLG3 addresses a shortcoming of NLG2 namely that the previous two words are not necessarily the best predictors for the next word. Instead, NLG3 conditions generation on syntactically related words by generating a syntactic dependency tree topd-down where each word is predicted in the context of ist syntactically related parent, grand parent and sibling. For NLG3 the training corpus is annotated with dependency trees.

The approach was test on flight descriptions with 26 attributes for the meaning representations and a training corpus of 6000 templates. The test set consisted of 1946 templates. The evaluation was human based and classified each output into perfectly acceptable / OK / Bad / No output. NLG2 and NLG3 were found to outperform the baseline by a large margin.

5.3. Generation by Inverting a Semantic Parser that uses Statistical Machine Translation

(Wong and Mooney 2007) inverts a semantic parser that uses SMT (Statistical Machine Translation) methods to map sentences into meaning representations.

The generation task is to find a sentence *e* such that (i) *e* is a good sentence and (ii) its meaning matches that of the input MR (meaning representation). To assess sentence quality, a language model is used. To assess the relation between sentence and meaning representation, an SMT based parsing model is used. Thus to find the best verbalisation *e* of a given meaning representation *f,* the approach maximises

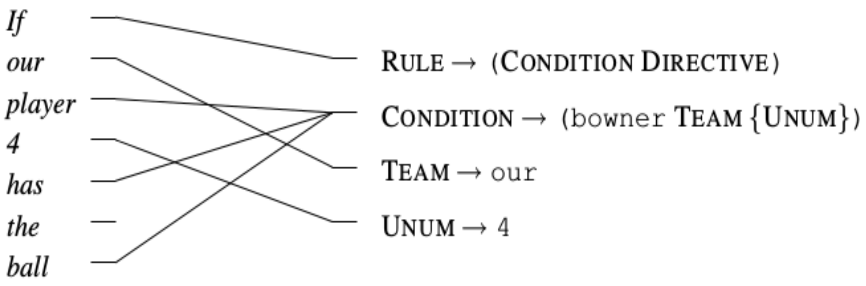
$$\Pr(e).\Pr(e|f) \approx max_{d \in D(f)} Pr(e(d)).Pr(d|e(d))$$
$$= max_{d \in D(f)} \frac{Pr(e(d)).exp \sum_i \lambda_i f_i(d)}{Z_\lambda(e(d))}$$

where *d(f)* is the set of derivations that are consistent with *f* and *e(d)* is the output sentence generated by *d*.

The derivations consistent with a given meaning representations are produced using a grammar based parser whose grammar is learned from a sentence aligned corpus of sentences and meaning representations. First Giza++ is used to obtain the best alignments from the training examples between word phrases and production rules describing the structure of the given meaning representation. Then SCFG (Synchronous Context Free Grammar) rules are extracted from these alignments. For instance, given the text and the meaning representation shown in (2a) and 2b) respectively

2a. *If our player 4 has the ball, then our player 6 should stay in the left side of our half.*
2b. ((bowner our {4})
(d our {6} (pos (left (half our)))))

a possible alignment and set of extracted rules would be as shown below.



$$\begin{aligned}
\textsc{Rule} &\rightarrow \langle \textit{if}\ \textsc{Condition}_{[1]}\ ,\ \textsc{Directive}_{[2]}\ .\ , \\
&\quad (\textsc{Condition}_{[1]}\ \textsc{Directive}_{[2]})\rangle \\
\textsc{Condition} &\rightarrow \langle \textsc{Team}_{[1]}\ \textit{player}\ \textsc{Unum}_{[2]}\ \textit{has the} \\
&\quad \textit{ball}\ ,\ (\texttt{bowner}\ \textsc{Team}_{[1]}\ \{\textsc{Unum}_{[2]}\})\rangle \\
\textsc{Team} &\rightarrow \langle \textit{our}\ ,\ \texttt{our}\rangle \\
\textsc{Unum} &\rightarrow \langle \textit{4}\ ,\ 4\rangle
\end{aligned}$$

A maximum entropy model is learned to define the conditional probability of derivations given an input sentence:

$$\Pr_\lambda(\mathbf{d}|\mathbf{e}) = \frac{1}{Z_\lambda(\mathbf{e})} \exp \sum_i \lambda_i f_i(\mathbf{d})$$

The feature functions f_i are the number of times each rule is used in a derivation. $Z_\lambda(e)$ is a normalizing factor and the model parameters λ_i are estimated using L-BFGS to maximise the conditional log likelihood of the training data.

The approach is evaluated on the RoboCup and on the GeoQuery corpora where the RoboCup corpus consists of 300 coach advice pieces aligned with a correponding CLANG meaning representation. The GeoQuery corpus consists of 880 questions manually translated to the function GeoQuery language. BLEU and NIST are used as well as a human evaluation targeting fluency and adequacy.

6. Joint Approaches

6.1. Comprehensive Probabilistic Generation

(Belz 2008) presents a probabilistic approach to generation where the generation task is seen as incrementally specifying a word string by probabilistically rewriting less specified representations (data) into more specified ones (linguistic representations and strings). The grammar is extracted from a paralled corpus of text and data e.g.,

Data: 1 SSW 16 20 - - 0600 2 SSE - - - - NOTIME 3 VAR 04 08 - - 2400
Forecast: SSW 16-20 GRADUALLY BACKING SSE THEN FALLING VARIABLE 4-8 BY LATE EVENING

First, a set of expansion rules is extracted semi-automatically from a parallel data-text corpus. Expansion rules are context free rules whose non terminals are terms of the form *f(b1 … bn)* with *b1 … bn* variables or constant and *f* an n-ary relation. They permit converting an input vector of numbers and symbols in steps to a set of natural language strings. For instance, in the domain of weather reports, a grammar of such expansion rules might support the following derivation:

$$Segment(2, -1, up, counterclock, slow, ssw, 22, 28, n, n, n) \stackrel{*}{\Rightarrow}$$

gradually backing SSW and gradually increasing 22-28
gradually backing and increasing SSW 22-28
backing/increasing gradually SSW 22-28
backing SSW increasing 22-28
backing gradually SSW 22-28

The first 4 arguments of the Segment relation encode contextual information: it is the second segment (2) but not the last (-1), that wind speed has increased compared to the preceding segment (up), that the wind direction has changed counter-clock wise (counterclock) and that the changes are gradual (slow). The last 7 arguments are part of an input vector of weather data.

The Figure below shows a fragment of the base generator which generates phrases describing gusts of winds.

$$Gusts(Nv_1, N_2, n) \rightarrow GustCore(Nv_1, N_2)$$
$$Gusts(Nv_1, N_2, ST) \rightarrow GustCore(Nv_1, N_2)\ GustPostMod(ST)$$

$$GustCore(Nv, n) \rightarrow GustTrans\ Num(Nv)$$
$$GustCore(Nv_1, Nv_2) \rightarrow GustTrans\ Num(Nv_1) - Num(Nv_2)$$

$$GustTrans \rightarrow \text{gusting}$$
$$GustTrans \rightarrow \text{gusts}$$
$$GustTrans \rightarrow \text{gusts to}$$
$$GustTrans \rightarrow \text{in gusts}$$
$$GustTrans \rightarrow \text{risk gusts to}$$
$$GustTrans \rightarrow \text{with gusts}$$
$$GustTrans \rightarrow \text{with gusts to}$$

$$GustPostMod(s) \rightarrow \text{in any showers}$$
$$GustPostMod(s) \rightarrow \text{in or near showers}$$
$$GustPostMod(s) \rightarrow \text{in showers}$$
$$GustPostMod(t) \rightarrow \text{in any thunderstorm}$$
$$GustPostMod(t) \rightarrow \text{in any thunderstorms}$$
$$GustPostMod(t) \rightarrow \text{in any thundery showers}$$

The type definitions define Nv to be any positive integer, *N* to be either *Nv* or unspecified *(n*) and *ST* to be *s, t* or *n* which encodes presence of showers, thunderstorms or the unspecified value. The topmost *Gusts* relation takes as arguments information about maximum gust speed, minimum gust speed and presence of showers/thunderstorms.

To extract the rules from the data, corpus sentences are analysed using a set of simple chunking rules that split wind statements into wind direction, wind speed, gust speed, gust statements, time expressions, transitions phrases, pre-modifiers and post-modifiers. This results in phrases about wind directions, wind speed etc which are used to define a CFG whose preterminal and lexical rules expands e.g., a non terminal representing a wind direction change to *backing*. Higher-level rules that combine sequences of preterminals into larger components are written manually taking care of text structuring, aggregation and elision.

For each sentence in the corpus, the extracted grammar is used to produce all possible derivation trees and a probability distribution over alternative rules is computed using frequency counts, add-1smoothing and standard maximum likelihood estimation. Thus if *c(N → α)* is the number obtained by adding 1/n to c for every occurrence of the rule in a set of derivation trees for a sentence where n is the total number of alternative derivations trees for the sentence, then

$$p(N \rightarrow \alpha) = \frac{c(N \rightarrow \alpha)}{\sum_{i:(N \rightarrow \alpha i) \in R} c(N \rightarrow \alpha i)}$$

During generation, inputs are rewritten using expansion rules and the probability distribution over expansion rules is used in one of three ways;

Greedy generation: the most likely rule is applied at each rewrite step.
Viterby generation: create a generation forest by applying all rules at each step, then do a Viterby search. This maximises the joint likelihood of all decisions taken.
Greedy roulette wheel generation: at each step select a rule according to a non uniform random distribution proportional to the likelihoods of expansion rules.

The approach was tested on an aligned corpus of 2123 instances corresponding to a total of 22985 words. It was evaluated using BLEU-4 and NIST-5 as well as a human evaluation. Greedy generation was found to be best and to be competitive with a hand crafted system.

6.2. A Hierarchical Discriminative Approach

(Angeli et al. 2011) presents an approach for generating text from database records where the generation process is broken up into a sequence of local decisions, arranged hierarchically and each trained discriminatively.

Given a corpus consisting of database records and text verbalising some of the information encoded in these records, they start by using the model of Liang et al. (2009) to automatically induce the correspondences between words in the text and the actual database records mentioned. The generation process is then structured into three sequential decisions: (i) record selection which determine which records in the input to talk about (content selection); (ii) field set decisions, which determine which fields of those records to mention: and (iii) template decisions which determine which words to use to describe the chosen fields (surface realisation).

Generation Process

for $i = 1, 2, \ldots$:
  choose a record $r_i \in \mathbf{s}$
  if $r_i = \text{STOP}$: return
  choose a field set $F_i \subset \text{FIELDS}(r_i.t)$
  choose a template $T_i \in \text{TEMPLATES}(r_i.t, F_i)$

Each of these decisions is made by a discriminative model whose features are designed to be domain-independent and which may depend on the current and all previous decisions (global features)[2]. Each record, field set and template (sequence of elements where each element is either a word or a field) is associated with a feature vector and at each step in the generation process the vector with maximal probability given the previous decision sequence is chosen (greedy search).

Feature Templates

Record	**R1**†	list of last k record types	$[\![r_i.t = * \text{ and } (r_{i-1}.t, \ldots, r_{i-k}.t) = *]\!]$ for $k \in \{1, 2\}$
	R2	set of previous record types	$[\![r_i.t = * \text{ and } \{r_j.t : j < i\} = *]\!]$
	R3	record type already generated	$[\![r_j.t = r_i.t \text{ for some } j < i]\!]$
	R4	field values	$[\![r_i.t = * \text{ and } r_i.v[f] = *]\!]$ for $f \in \text{FIELDS}(r_i.t)$
	R5†	stop under language model (LM)	$[\![r_i.t = \text{STOP}]\!] \times \log p_{\text{LM}}(\text{STOP} \mid \text{previous two words generated})$
Field set	**F1**†	field set	$[\![F_i = *]\!]$
	F2	field values	$[\![F_i = * \text{ and } r_i.v[f] = *]\!]$ for $f \in F_i$
Template	**W1**†	base/coarse generation template	$[\![h(T_i) = *]\!]$ for $h \in \{\text{BASE}, \text{COARSE}\}$
	W2	field values	$[\![h(T_i) = * \text{ and } r_i.v[f] = *]\!]$ for $f \in F_i, h \in \{\text{BASE}, \text{COARSE}\}$
	W3†	first word of template under LM	$\log p_{\text{LM}}(\text{first word in } T_i \mid \text{previous two words})$

[2] This is in contrast with eg Belz 2008's approach where decisions must decompose locally according to a tree.

The parameters of the model (feature weights) are learned by maximising the conditional likelihood of the training data which consists of pairs of database records and text which have been aligned and where text has been generalized either to the corresponding field name or to those field names that are licensed by some manually defined trigger pattern.

The approach is evaluated on three domains namely, robocup games (1919 entries), weather forecasts (SumTime-Meteo corpus, 469 entries) and WeatherGov (29528 entries) using BLEU-4 and a human-based evaluation.

6.3. Generation with a PCFG represented as an Hypergraph

(Konstas and Lapata 2012) describe an end-to-end approach to generation in which a PCFG is learned captures the structure of a database **d** with records and fields as intermediate non-terminals, and words *w* (from the associated text) as terminals. The grammar is represented as a weighted directed hypergraph and the weights are computed using a dynamic program similar to the inside-outside algorithm.

Given a trained grammar and a database fragment, the model generates text by compiling a hypergraph specific to the test input (database elements) and decoding the hypergraph via cube pruning. That is generation finds the text *w* which maximises:

$$argmax_w P(w|d) = argmax_w P(\boldsymbol{w}).P(d|w)$$

where *P(w|d)* is the decoding likelihood for a sequence of word w. *P(w)* is a measure of the quality of each output (given by a language model) and *P(w|d)* is the posterior of the output for database d.

The approach is tested and evaluated on three domains namely, weather forecast generation (WeatherGov), air travel (ATIS) and soccer commentaries (Robocup). It is shown to outperform 2 previous approaches (Angeli et al. 2010, Kim and Mooney 2010) in terms of BLEU scores and in a human evaluation for fluency and semantic correctness.

The approach is refined in (Konstas and Lapata 2013) to produce documents plans which are either induced from the data or from an RST discourse parser. This is done by replacing the grammar rules describing the chaining of database records by discourse rules describing how records group into sentences and how sentences group into a document.

In the first approach (discourse rules induced from data), the new discourse rules are extracted from the training data as follows. First datbase records and words are aligned using Liang et al's (2009) unsupervised model. The aligned record tokens are them mapped to their corresponding type, adjacent words with the same record type are merged and text is segmented on punctuation. Next the corresponding discourse tree is created and binarized. That is words are grouped to reflect the record they verbalise, records are grouped to reflect the sentence they cover and sentences are attached to the root discourse node. Rule weights are counted on the resulting treebanks and rules with frequency less than 3 are discarded.

In the second RST type approach, each record is mapped to an EDU (elementary discourse unit) and an existing discourse parser is used to parse the resulting corpus and extract rule weights.

The approach is evaluated on the Weather Gov and the WinHelp corpus and shown to outperform the previous version of the system by a wide margin on both datasets.

6.4. Integer Linear Programming for Content Selection, Lexicalisation and Aggregation

(Lampouras and Androutsopoulos 2013) presents an ILP model of content selection, lexicalisation and aggregation that generates text from ontology. Compared to a traditional pipeline architecture, the ILP approach permits avoiding greedy decisions. The approach was tested on two ontologies and shown to produce more compact text than a standard pipeline approach with no deterioration in the perceived quality of the generated texts.

For each fact (OWL axiom) in the knowledge base, a set of alternative sentence plans is available where a sentence plan is a sentences containing slots annotated with instructions specifying how to fill slots. For instance, the fact:

(exhibit12, foundIn, athens)

is associated with the sentence plan

[ref (S)] [find past] [in] [ref (O)]

where square brackets denote slots, ref (S) and ref (O) are instructions requiring referring expressions for S and O in the corresponding slots, and “find past ” requires the simple past form of “find”. In our example, the sentence plan would lead to a sentence like “Exhibit 12 was found in Athens”.

The slots and the individuals, classes or datatype values they refer are called “elements”.

Aggregation rules (Dalianis, 1999) are used that operate on sentence plans and usually lead to shorter texts e.g.,

> Bancroft Chardonnay is a kind of Chardonnay. It is made in Bancroft
> ⇒Bancroft Chardonnay is a kind of Chardonnay made in Bancroft

The ILP model makes use of the following variables:

- F = $\{f_1,\ldots,f_n\}$is the set of facts to be verbalised
- $s_1,\ldots,s_n$are disjoints subsets of F each containing 0 to n facts with $m \preccurlyeq n$
- a_i = 1 if $f_i$is selected, 0 otherwise
- l_{ijk}= 1 if sentence plan $p_{ik}$is used to express fact $f_i$and $f_i$is in subset s_j, 0 otherwise
- b_{tj} = 1 if element e_t is used in subset s_j, 0 otherwise

Given a set of facts to be verbalised, the objective function maximises the total importance of selected facts (each fact is assumed to be annotated with an importance score) and minimises the

number of distinct elements in each subset i.e., the approximate lengths of the corresponding aggregated sentence. In addition, well formed solutions must obey the following five constraints:

1. For each selected fact, only one sentence plan in only one subset is selected
2. If a sentence plan is selected in a given subset, then all elements of that sentence plan are also selected in that subset
3. If an element is used in a subset than at least one of the sentence plans involving that element mus also be selected in that subset
4. The number of elements that a subset may contain is limited to a maximum allowed number (this limits the maximum lengths of a sentence).
5. Facts from different sections are not placed in the same subset (it is assumed that each relation has been manually mapped to a single topical section).

6.5. A Ranking Framework for Planning and Realisation

(Kondadadi et al. 2013) presents a statistical approach to generation in which automatically induced sentence templates are ranked for position in the output text using an SVM (Support Vector Machine).

Corpus sentences are processed using the deep semantic parser Boxer. The DRS (Discourse Representation Structure) produced by Boxer and the domain specific named entity tags produced by domain specific named entity taggers are used to associate each sentence in the corpus with a meaning representation and the corresponding template. For instance, the sentence

> *Mr. Mitsutaka Kambe has been serving as Managing Director of the 77 Bank, Ltd. since June 27, 2008.*

is associated with the conceptual meaning:

> SERVING | TITLE | PERSON | COMPANY | DATE

and with the template

> [person] has been serving as [title] of the [company] since [date]

Next templates are clustered into conceptual units using k-means clustering and similarity between templates so that each conceptual unit is associated with a set of templates and conversely, each template is associated with a conceptual unit.

A training corpus is then built which aligns each corpus sentence S with a ranked list of templates where the rank of each template t is determined by its Levenshtein edit distance to the template corresponding to S. In addition templates are associated with a feature vector using the features listed below.

- **CuId given position**: This is a binary feature where the current CuId is either the same as the most frequent CuId for the position (1) or not (0).
- **Overlap of named entities**: Number of common entities between current CuId and most likely CuId for the position
- **Prior template**: Probability of the sequence of templates selected at the previous position and the current template (iterated for the last three positions).
- **Prior CuId**: Probability of the sequence of the CuId selected at the previous position and the current CuId (iterated for the last three positions).
- **Difference in number of words**: Absolute difference between number of words for current template and average number of words for the CuId
- **Difference in number of words given position**: Absolute difference between number of words for current template and average number of words for CuId at given position
- **Percentage of unused data**: This feature represents the portion of the unused input so far.
- **Difference in number of named entities**: Absolute difference between the number of named entities in the current template and the average number of named entities for the current position
- **Most frequent verb for the position**: Binary valued feature where the main verb of the template belongs to the most frequent verb group given the position is either the same (1) or not (0).
- **Average number of words used**: Ratio of number of words in the generated text so far to the average number of words.
- **Average number of entities**: Ratio of number of named entities in the generated text so far to the average number of named entities.
- **Most likely CuId given position and previous CuId**: Binary feature indicating if the current CuId is most likely given the position and the previous CuId.
- **Similarity between the most likely template in CuId and current template**: Edit distance between the current template and the most likely template for the current CuId.
- **Similarity between the most likely template in CuId given position and current template**: Edit distance between the current template and the most likely template for the current CuId at the current position.

The weight associated with each feature is learned using a linear kernel for a ranking SVM.

At generation time, the input consists of a set of input data, a template bank and the SVM ranking model. Templates are then selected and filled iteratively by chosing the template that rank highest at each step of the generation process and filling its slots with matching entity tags from the input data.

The approach is tested on two domains: corporate officer and director biographies and oil rig weather reports from the SumTime Meteo corpus. BLEU-4 and Meteor are used as well as a human based evaluation. The baseline simply chooses the most frequent conceptual unit at the given position and the most likely template for the conceptual unit.

7. Conclusion

While there are many possible ways of handling the various issues arising when generating text, one common issue is how to relate natural language words and phrases to data items. Indeed, most of the approaches described in this deliverable incorporate an alignment step in which words in text are aligned with data items. This step is more or less complex depending on the available text and data corpora and on the model used.

(Belz 2008, Konstas and Lapata 2013, Ratnparkhi 2000, Wong and Mooney 2007) all explore a setting in which a parallel corpus is available: short texts are aligned with database fragments in so called scenarii. Given this parallel corpus, the alignment between words and data can be learned using various techniques. (Belz 2008) uses a mixture of hand written rules and automatically learned rules to learn a probabilistic grammar that rewrites data vectors into text. (Konstas and Lapata 2013) automatically induce a probabilistic context free grammar from aligned database and text fragments. (Ratnaparkhi 2000) learns an incremental generation model from a corpus of generation templates i.e., of sentences in which values of interests (semantic values) have been replaced with corresponding attributes. His generation system makes use of 3 models: a first model to choose the most frequent template in the training data that corresponds to a set of attributes, a second model to choose the most likely word given the words already generated and a third model to improve syntactic well formedness. In his approach, the alignment between text and data is given by the annotated templates and used to train models that guide choices at generation time. (Wong and Mooney 2007) train a statistical machine translation model on an aligned corpus of text and semantic rule describing the semantic structure of the input data.

Other approaches investigate how to relate text and data in a context where no parallel data/text corpus is available. (Duboue and McKeown 2003) use clustering, cross entropy and a decision tree algorithm to learn which semantic data is actually expressed in the text. (Ell and Harth 2014) induce verbalization templates from Wikipedia which can verbalise sets of DBPedia RDF triples using matching and frequent maximal subgraph extraction. (Cimiano et al 2013) presents a similar approach where sentences that contain mentions matching entities occurring in an RDF triples are extracted from a large corpus and used to create sentence templates. (Kondadadi et al 2013) also automatically induce sentence templates from data but additionally learn a ranker to determine which template is most appropriate given the input.

Thus alignments can be learned using either parallel or comparable data/text corpora. In ModelWriter, models and text may give rise to both situations.

In Use Case UC-FR1 (Synchronisation between models and documentation, OBEO – Sirius Product) for instance, there is no direct alignment of text and data. In this usecase the synchronization mechanism relies on the detection and creation of “links” .between elements in the model or the java code and elements in the documentation. In this case, the alignment techniques used for generation from comparable corpora can be used to support Obeo’s synchronization system. We will therefore investigate how the detailed alignment technique described in (Ell and Harth) can be adapted to support the semantic annotation of Sirius documentation by model and Java code items and in this way, automatically produce the links that are required by the synchronization mechanism to signal to the user which parts of the documentation might need updating. The alignment tool developed could also be used in usecase

UC-FR4 to synchronise elements of the SIDP normalized rules with the SIDP documents thus facilitating access to the relevant portion of the SIDP document from the given rule.

In contrast, in use case UC-FR4 (Synchronisation of regulation documentation with a design rule repository), because RDF data will be produced from the text by semantic parsing, parallel data will be available. We will therefore explore data-to-text alignment techniques developed for parallel data/text corpora taking inspiration from (Belz 2008, Konstas and Lapata 2013, Ratnparkhi 2000).

As shown in the approaches just mentioned, once the data is aligned, lexicalizations and sentence templates can be extracted. and used for surface realization. However many choices remain to be made including, content selection, content planning, referring expression generation, and surface realization. In the context of ModelWriter, content selection is less an issue as generation is meant to reflect changes in the model. Content selection can therefore be determined by the changes made. i.e., the content to be verbalized is the content affected by the change. Content planning is also not a major issue as the changes to be verbalized will usually be local. In short, most of the work will consist in developing a surface realizer which produces high quality text from the lexicalizations induced through data-to-text alignment. To this end we will explore the development of a joint model similar to those described in (Belz 2008, Angeli et al. 2011, Konstas and Lapata 2012, Lampouras and Androutsopoulos 2013, Kondadadi 2013). As explained in the FPP, the aim is WP2 is to develop a reversible processor which can both parse text into models and generate text from modes. Consequently, for generation, we will take as a starting point the RDF representations produced by the semantic parser developed for the use case UC-FR4. In a first step, the aim will be to regenerate the sentences from which the RDF data was derived. Since parallel data will be available we will start by investigating the model described in Belz 2008. That is we will learn a probabilistic context free grammar describing how data rewrite into strings. In a second step, we will evaluate the model on unseen data i.e. on sets of RDF triples that are not produced by the semantic parser. This will be useful for instance, to verbalise answers to queries formulated on the knowledge base produced by the semantic parser from the normalized SIDP rules.

8. Bibliography

Androutsopoulos, Ion, Gerasimos Lampouras, and Dimitrios Galanis. "Generating natural language descriptions from OWL ontologies: the NaturalOWL system." *Journal of Artificial Intelligence Research* (2013): 671-715.

Angeli, Gabor, Percy Liang, and Dan Klein. "A simple domain-independent probabilistic approach to generation." *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* 9 Oct. 2010: 502-512.

Bangalore, Srinivas, and Owen Rambow. "Using TAG, a tree model, and a language model for generation." *In Proceedings of the 1st International Natural Language Generation Conference* 2000.

Barzilay, Regina, and Mirella Lapata 2005. "Collective content selection for concept-to-text generation." *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 2005, pages 331-338.

Belz, Anja. "Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models." *Natural Language Engineering* 14.04 (2008): 431-455.

Bouayad-Agha, Nadjet, Gerard Casamayor, and Leo Wanner 2011. "Content selection from an ontology-based knowledge base for the generation of football summaries." *Proceedings of the 13th European Workshop on Natural Language Generation* 28 Sep. 2011: 72-81.

Bouayad-Agha, N., Casamayor, G., Rospocher, M., Saggion, H., Serafini, L., and Wanner, L. From ontology to NL: Generation of multilingual user-oriented environmental reports. In Proceedings of the 17th International Conference on Appications of Natural Language Processing to Information Systems (NLDB 2012). Lecture Notes in Computer Science series, vol. 7337 (Groningen, NL, 2012).

Cahill, Aoife, and Josef Van Genabith. "Robust PCFG-based generation using automatically acquired LFG approximations." *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* 17 Jul. 2006: 1033-1040.

Carroll, John, and Stephan Oepen. "High efficiency realization for a wide-coverage unification grammar." *Natural Language Processing–IJCNLP 2005* (2005): 165-176.

Cimiano, Philipp et al. "Exploiting ontology lexica for generating natural language texts from RDF data." (2013).

Dale, Robert. "Cooking up referring expressions." *Proceedings of the 27th annual meeting on Association for Computational Linguistics* 26 Jun. 1989: 68-75.

Dale, Robert, and Ehud Reiter. "Computational interpretations of the Gricean maxims in the generation of referring expressions." *Cognitive science* 19.2 (1995): 233-263.

Demir, Seniz, Sandra Carberry, and Kathleen F McCoy. "A discourse-aware graph-based content-selection framework." *Proceedings of the 6th International Natural Language Generation Conference* 7 Jul. 2010: 17-25.

Dimitromanolaki, Aggeliki, and Ion Androutsopoulos. "Learning to order facts for discourse planning in natural language generation." *arXiv preprint cs/0306062* (2003).

Duboue, Pablo A, and Kathleen R McKeown. "Empirically estimating order constraints for content planning in generation." *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics* 6 Jul. 2001: 172-179.

Duboue, Pablo A, and Kathleen R McKeown. "Content planner construction via evolutionary algorithms and a corpus-based fitness function." *Proceedings of INLG 2002* Jul. 2002: 89-96.

Duboue, Pablo A, and Kathleen R McKeown. "Statistical acquisition of content selection rules for natural language generation." *Proceedings of the 2003 conference on Empirical methods in natural language processing* 11 Jul. 2003: 121-128.

Galanis, Dimitrios, and Ion Androutsopoulos. "Generating multilingual descriptions from linguistically annotated OWL ontologies: the NaturalOWL system." *Proceedings of the Eleventh European Workshop on Natural Language Generation* 17 Jun. 2007: 143-146.
Gargett, Andrew et al. "The GIVE-2 Corpus of Giving Instructions in Virtual Environments." *LREC* May. 2010.

Garoufi, Konstantina, and Alexander Koller. "Combining symbolic and corpus-based approaches for the generation of successful referring expressions." *Proceedings of the 13th European Workshop on Natural Language Generation* 28 Sep. 2011: 121-131.

Kan, Min-Yen, and Kathleen McKeown. "Corpus-trained text generation for summarization." (2002).

Kelly, Colin, Ann Copestake, and Nikiforos Karamanis. "Investigating content selection for language generation using machine learning." *Proceedings of the 12th European Workshop on Natural Language Generation* 30 Mar. 2009: 130-137.

Kim, Joohyun, and Raymond J Mooney. "Generative alignment and semantic parsing for learning from ambiguous supervision." *Proceedings of the 23rd International Conference on Computational Linguistics: Posters* 23 Aug. 2010: 543-551.

Krahmer, Emiel, and Kees Van Deemter. "Computational generation of referring expressions: A survey." *Computational Linguistics* 38.1 (2012): 173-218.

Ell, Basil, and Andreas Harth. "A language-independent method for the extraction of RDF verbalization templates." *INLG 2014* (2014): 26.

Howald, Blake, Ravi Kondadadi, and Frank Schilder. "Domain adaptable semantic clustering in statistical nlg." *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)* 2013: 143-154.

Schilder, Frank, Blake Howald, and Ravi Kondadadi. "Gennext: A consolidated domain adaptable nlg system." *Proceedings of the 14th European Workshop on Natural Language Generation* 8 Aug. 2013: 178-182.

Kondadadi, Ravi, Blake Howald, and Frank Schilder. "A Statistical NLG Framework for Aggregated Planning and Realization." *ACL (1)* 6 Aug. 2013: 1406-1415.

Konstas, Ioannis, and Mirella Lapata. "Unsupervised concept-to-text generation with hypergraphs." *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* 3 Jun. 2012: 752-761.

Konstas, Ioannis, and Mirella Lapata. "Inducing Document Plans for Concept-to-Text Generation." *EMNLP* 2013: 1503-1514.

Lampouras, Gerasimos, and Ion Androutsopoulos. "Using Integer Linear Programming in Concept-to-Text Generation to Produce More Compact Texts." *ACL (2)* 4 Aug. 2013: 561-566.

Liang, Percy, Michael I Jordan, and Dan Klein. "Learning semantic correspondences with less supervision." *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1* 2 Aug. 2009: 91-99.

McCrae, John, Dennis Spohr, and Philipp Cimiano. "Linking lexical resources and ontologies on the semantic web with lemon." *The Semantic Web: Research and Applications* (2011): 245-259.

Mellish, Chris, and Xiantang Sun. "Natural language directed inference in the presentation of ontologies." *ENLG, Aberdeen, Scotland, August* (2005).

Mellish, Chris, and Jeff Z Pan. "Natural language directed inference from ontologies." *Artificial Intelligence* 172.10 (2008): 1285-1315.

Narayan, Shashi, and Claire Gardent. "Structure-driven lexicalist generation." *24th International Conference in Computational Linguistics (COLING)* 8 Dec. 2012: 100-113.

Page, Lawrence et al. "The PageRank citation ranking: Bringing order to the web." (1999).

Ratnaparkhi, Adwait. "Trainable methods for surface natural language generation." *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference* 29 Apr. 2000: 194-201.

Reiter, Ehud, Robert Dale, and Zhiwei Feng. *Building natural language generation systems*. Cambridge: Cambridge university press, 2000.

Schapire, Robert E, and Yoram Singer. "BoosTexter: A boosting-based system for text categorization." *Machine learning* 39.2-3 (2000): 135-168.

Walter, Sebastian, Christina Unger, and Philipp Cimiano. "A corpus-based approach for the induction of ontology lexica." *Natural Language Processing and Information Systems* (2013): 102-113.

White, Michael, Rajakrishnan Rajkumar, and Scott Martin. "Towards broad coverage surface realization with CCG." *Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+ MT)* 11 Sep. 2007: 267-276.

Wong, Yuk Wah, and Raymond J Mooney. "Generation by Inverting a Semantic Parser that Uses Statistical Machine Translation." *HLT-NAACL* 2007: 172-179.

Zarrieß, Sina, and Jonas Kuhn. "Combining Referring Expression Generation and Surface Realization: A Corpus-Based Investigation of Architectures." ACL (1) 2013: