The PERSONAGE Generator User Manual

François Mairesse

October 25, 2009

Table of Contents

Chapter 1

Introduction

PERSONAGE is a natural language generator modelling findings from psychological studies to project various personality traits through language. PERSONAGE implements various generation paradigms: (1) rule-based generation, (2) overgenerate and select and (3) generation using parameter estimation models—a novel approach that learns to produce recognisable variation along meaningful stylistic dimensions without the computational cost incurred by overgeneration techniques. PERSONAGE was shown to produce recognisable personality, according to human judges [Mairesse and Walker, 2007, 2008, Mairesse, 2008].

This PERSONAGE distribution contains resources for two domains: the MATCH restaurant recommendation domain on which the SPaRKy sentence planner was evaluated (MATCH), as well as the Cambridge Tourist Information domain (CamInfo). The following sections assume that PERSONAGE is used for the MATCH domain (see configuration files).

The next Chapter describes how to install and run Personage in various modes. Chapter 3 presents the levels of input that can be specified to generate an utterance: from a high-level communicative goal to low-level syntactic structures. It also contains details about the configuration file parameters and the package organisation. Chapter 5 details each generation component of the PERSONAGE base generator, which are pipelined to gradually convert a high-level representation of the utterance into a surface string. While the PERSONAGE base generator requires traditional generation parameters, Chapter 6 presents the different generation paradigms that can be used to control the personality of the base generator: (1) rule-based generation (PERSONAGE-RB, (2) overgenerate and select (PERSONAGE-OS) and (3) parameter estimation models (PERSONAGE-PE). Finally, Chapter 7 provides detailed examples on how to run the generator with different inputs and generation methods.

Chapter 2

Quick start

One should follow these steps in order to install and run PERSONAGE:

1. Contact Cogentex and ask them for a RealPro serial number for research purposes, and modify the file `lib/RealPro-2.3/RealPro.properties` accordingly.

2. Extract the personage archive (e.g. in `PERSONAGE_DIR`), and make sure you have JDK 1.6 or above installed, and your `JAVA_HOME` environment variable pointing to its root directory, i.e. such that the java interpreter can be reached using `$JAVA_HOME/bin/java`.

3. Modify the `PERSONAGE_DIR/PersonageMATCH.properties` if necessary, but the properties should already point to the correct `libraries/resources` contained in the archive.

4. If under Unix/Linux, you can compile Personage by running 'make clean' followed by 'make' in the root directory. You might want to edit the Makefile if you want to use different libraries that the ones in the archive. If you are running Windows, you would have to use the Windows version of make, and change the classpath separator in the Makefile to SEP = ; (not tested).

5. Make `PERSONAGE_DIR/PersonageExample` executable (e.g. `chmod 755 PERSONAGE_DIR/PersonageExample`). Under Windows, need to convert the shell script into a batch file.

6. You can test your installation by generating an example utterances with different styles by running `./PersonageExample [-OPTION]` in the `PERSONAGE_DIR` directory. The example Java code is in `PERSONAGE_DIR/src/generator/ExampleMATCH.java`. The rule-based version is the default, but the script takes the following options:

    - `-pe`: train and use parameter estimation models (see ACL 2008 paper)

- `-seqpe`: load and use sequential parameter estimation models predicting each parameter based on the stylistic dimensions AND the actual values of the previous parameters in the pipeline.
- `-rand`: find optimal parameters by searching over the input space of overgeneration selection models (takes longer).

7. You can run the PERSONAGE graphical interface by running `PERSONAGE_DIR/PersonageGUI` (rule-based version). The interface allows you to see how changing individual parameters affects the generated output. It also allows new XML parameter files to be loaded and tested.

Chapter 3

Input structure

As PERSONAGE generates language in the information presentation domain, it can take as an input high-level communicative goals, represented as a content plan. Predefined content plans include the *recommendation* of an entity, and a *comparison* of multiple entities. Moreover, PERSONAGE also accepts lower-level inputs such as a list of syntactic structures, although higher level generation parameters are then discarded, resulting in a smaller range of variation. PERSONAGE is currently implemented for the restaurant domain, entities are thus specific restaurants retrieved from a database.

3.1 Content plan

PERSONAGE accepts high-level communicative goals for presenting information, represented as a *content plan*.[1]

The content plan combines together propositions expressing information about individual attributes using *rhetorical relations* from Mann and Thompson's Rhetorical Structure Theory (RST; 1988). RST is typically used to study the coherence of texts, by recursively defining how multiple spans of text relate to each other. Whenever a span of text is more essential for conveying the desired information, it is referred to as the *nucleus* of the relation (N), whereas the other text span is defined as the *satellite* (S). Two types of communicative goals are currently supported in PERSONAGE: *recommendation* and *comparison* of restaurants. Figure 3.1 shows an example content plan tree for a recommendation. Each recommendation content plan contains a claim (nucleus) about the overall quality of the selected restaurant(s), supported by a set of satellite propositions describing their attributes. The propositions—the leaves in the content plan tree—are assertions labelled *assert-attribute(selection name)* in Figure 3.1. Recommendations are characterised by a JUSTIFY rhetorical relation associating the claim with all the other propositions, which are linked together through an INFER relation. The JUSTIFY relation is a

[1] Content plans are sometime referred to as *text plans* in the literature.

mononuclear rhetorical relation in which the satellite supports the speaker's right to express the nucleus. The INFER relation is a multinuclear relation expressing related information without specific constraints.[2]

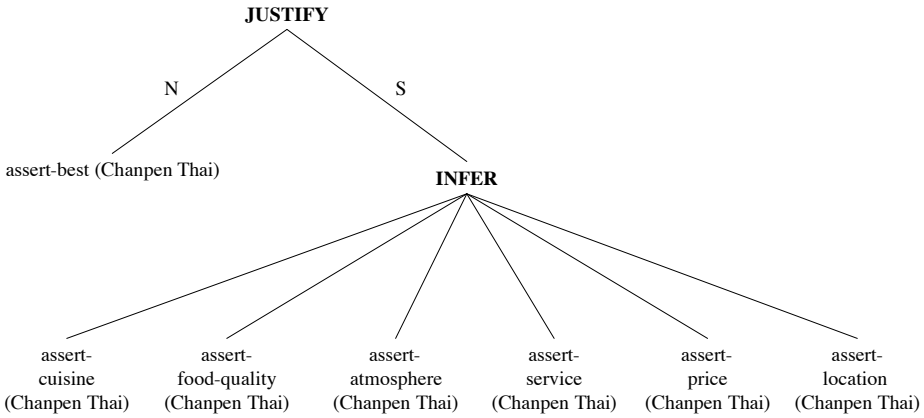


Figure 3.1: An example content plan tree for a recommendation for Chanpen Thai, using all the restaurant attributes. N = nucleus, S = satellite.

In comparisons, the attributes of multiple restaurants are compared using the CONTRAST multinuclear rhetorical relation. This relation combines propositions describing the same attributes for different restaurants, joined together through an INFER relation. An example content plan tree for a comparison between two restaurants is illustrated in Figure 3.2.

PERSONAGE currently provides support for generating both *recommendation* and *comparison* of multiple entities from a restaurant database. They can be generated using methods from the Personage.java interface (see ExampleMATCH.java):

```
// Create selection and attributes for one restaurant in the database
Map<String,Set<TextPlanBuilder.DBAttribute>> restaurantGroup
    = new LinkedHashMap<String,Set<TextPlanBuilder.DBAttribute>>(1);
restaurantGroup.put("Trattoria Rustica",
    new LinkedHashSet<TextPlanBuilder.DBAttribute>());
restaurantGroup.get("Trattoria Rustica").add(
    TextPlanBuilder.DBAttribute.FOOD_QUALITY);
restaurantGroup.get("Trattoria Rustica").add(
    TextPlanBuilder.DBAttribute.SERVICE);

// Create recommendation plan
 Document contentPlan
    = tpBuilder.createRecommendationPlan(restaurantGroup, true);
```

[2]The INFER relation is similar to the JOINT relation in the RST literature.

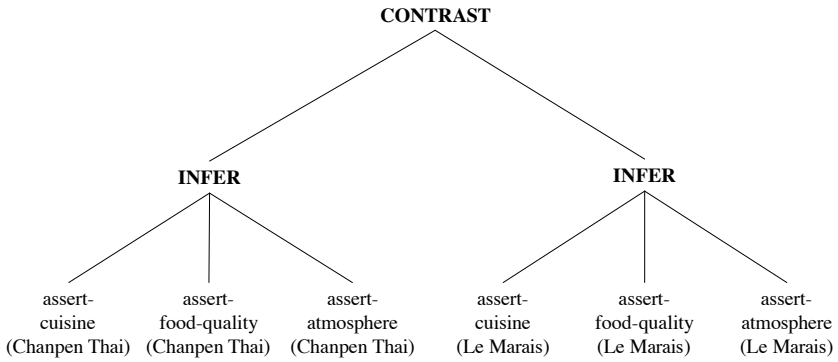


Figure 3.2: An example content plan tree for a comparison between Chanpen Thai and Le Marais, using three attributes. All relations are multinuclear.

These methods respectively generate the content plan trees in Figures 3.1 and 3.2. Content plan trees are represented in xml DOM format as in Figure 3.3 for a recommendation for 'Trattoria Rustica', which contains a specification of the RST tree under the `<rstplan>` node, followed by list specifying the content of each proposition referred to in the RST tree (attribute `id`). The proposition element's `dialogue\_act` attribute determines the syntactic template that will be used by referring to the stem of a file containing a Deep Syntactic Structure in the input/dss directory. For example, the file `assert-reco-best.xml` referred to by the first proposition in Figure 3.3 defines a template expressing the fact that the selected restaurant is the best. The `attribute` elements instantiate variables in the syntactic template, e.g. with the selection's name (e.g. `name="SELECTION" value="Trattoria Rustica"`), the attribute's name (e.g. `name="ATTR" value="FOOD_QUALITY"`) or a scalar value indicating the polarity of the attribute (e.g. `name="MOD_REAL" value="20"`, values are on a scale from 0 to 30 apart from the `price` attribute).

Users can generate their own content plans by following the structure in Figure 3.3. Once an xml content plan has been created, it can be generated using the `generate` methods in the Personage.java interface (see `ExampleMATCH.java`).

```
// Create content plan document (see above)
// (the content plan can also be loaded from a file using an XML parser)
Document contentPlan = tpBuilder.createRecommendationPlan(restaurantGroup, true);

// Create generator
Personage generator = new PersonageRuleBased(configurationFile);

// Specify target scores (e.g. high extrovert)
Map<VariationDimension,Double> extravertTarget =
    new LinkedHashMap<VariationDimension, Double>(1);
extravertTarget.put(VariationDimension.EXTRAVERSION, 7.0);
```

```
// Generate utterance from content plan
String utterance = generator.generate(contentPlan, extravertTarget);
```

3.2 Deep syntactic structure

In case high-level content information is not available, PERSONAGE can also modify syntactic structures directly, by ignoring higher-level parameters. This feature can be useful if one wants to modify the style of an existing dialogue from its syntactic representation. However, it is likely that the naturalness of the output utterances will be inferior to utterances generated from a content representation, since there is no guarantee that syntactic transformation rules will be applied successfully on arbitrary syntactic structures (e.g. questions). Structures representing assertions such as *NP has ADJ NP* or *NP is ADJ* are the most likely to be transformed without introducing ungrammatical constructions. Details about the syntactic transformation can be found in Section **??**.

PERSONAGE provides support for the Deep Syntactic Structure formalism (DSynts) introduced by Melčuk [1988], which can be realised by the RealPro realiser [Lavoie and Rambow, 1997]. Example DSyntS templates in xml format can be found in the `input/dss` directory, or in the RealPro user manual. To run Personage on a list of DSyntS, they must be first parsed into a `Document` object by an XML parser and added to an `Utterance` object, which can then be processed using the `Personage.transformSyntax` method:

```
public String generateDSyntS(File dsyntsXMLFile,
                             Map<VariationDimension,Double> target,
                             Personage generator) {

    // Load parser (e.g. Xerces) and parse DSyntS file
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    DocumentBuilder parser = dbf.newDocumentBuilder();
    Document doc = parser.parse(dsyntsXMLFile);

    // Add the root node of each DSyntS to a list
    List<Element> dsyntsList = new ArrayList();
    NodeList dsyntsNodes = doc.getElementsByTagName("DSYNTS");
    for (int i=0; i < dsyntsNodes.getLength(); i++) {
        dsyntsList.add((Element) dsyntsNodes.item(i));
   }

    // Set the target personality scores
    generator.setVariationDimensions(target);
```

```
<textplan name="recommend-Trattoria_Rustica">
<speechplan>
    <rstplan>
        <relation name="justify">
            <proposition id="1" ns="nucleus" />
            <proposition id="2" ns="satellite" />
        </relation>
        <relation name="justify">
            <proposition id="1" ns="nucleus" />
            <proposition id="3" ns="satellite" />
        </relation>

        ...
    </rstplan>
    <proposition dialogue_act="assert-reco-best" id="1">
        <attribute arg="" name="ATTR" value="BEST" />
        <obj>
            <attribute arg="X3" name="NUM" value="1" />
            <attribute arg="X1" name="TYPE" value="restaurant" />
            <attribute arg="X2" name="SELECTION" value="Trattoria Rustica" />
        </obj>
    </proposition>
    <proposition dialogue_act="assert-reco-food_quality" id="2">
        <attribute arg="" name="MOD_REAL" value="20" />
        <attribute arg="" name="ATTR" value="FOOD_QUALITY" />
        <attribute arg="X4" name="MOD_LEX" value="GOOD" />
        <attribute arg="" name="MOD_WEIGH" value="" />
        <obj>
            <attribute arg="X1" name="TYPE" value="restaurant" />
            <attribute arg="X2" name="SELECTION" value="Trattoria Rustica" />
        </obj>
    </proposition>
    <proposition dialogue_act="assert-reco-service" id="3">
        <attribute arg="" name="MOD_REAL" value="18" />
        <attribute arg="" name="ATTR" value="SERVICE" />
        <attribute arg="X4" name="MOD_LEX" value="DECENT" />
        <attribute arg="" name="MOD_WEIGH" value="" />
        <obj>
            <attribute arg="X1" name="TYPE" value="restaurant" />
            <attribute arg="X2" name="SELECTION" value="Trattoria Rustica" />
        </obj>
    </proposition>

    ...
</speechplan>
```

Figure 3.3: Example content plan tree for a recommendation in XML format.

```
    // Create an utterance and generate it
    return generator.transformSyntax(new Utterance(dsyntsList));
}
```

When used in this way, the content parameters of PERSONAGE are ignored, thus it is important to make sure that the input DSyntS do not express any style or personality that might conflict with the generator's target scores, in order to ensure that the output utterance does not convey inconsistent cues.

3.3 Directory structure

The PERSONAGE distribution is structured as follows:

- `lib`: various libraries required by PERSONAGE, e.g. RealPro, JWNL (modified version), Weka 3.5.5.
- `src`: Java source files.
- `bin`: binary Java classes.
- `doc`: Javadoc and manual.
- `input/db`: XML database of restaurants for the MATCH domain.
- `input/textplans`: SPaRky text plans in XML format.
- `resources/dsynts/match`: generation dictionary in XML DSyntS format for the MATCH restaurant recommendation domain.
- `resources/dsynts/markers`: database of syntactic pragmatic marker insertion patterns.
- `resources/lexical/wordnet`: WordNet 2.0 linux distibution (should also be loaded by JWNL under Windows).
- `resources/lexical/wn_senses`: sense-tagged word list following the format `WORD#PART_OF_SPEECH#SENSE_NUMBER`. Only sense-tagged words can be substituted.
- `resources/lexical/frequencies`: BNC frequency counts.
- `resources/lexical/mrc`: MRC Psycholinguistic database.
- `resources/lexical/adjectives`: adjectives grouped by polarity groups.
- `resources/lexical/attributes`: CamInfo mapping of slot values to lexemes.

- resources/lexical/verbocean: distribution of the VerbOcean database of verb relations.
- resources/parameters/handcrafted/bigfive: XML files containing handcrafted parameter values for each extreme of each Big Five traits, organised by NLG modules (see PhD thesis). As parameters can also be model features/classes, they can also be represented by a flat string using the TYPE_COMPONENT:PARAMETERTYPE:(TRUE)?PARAMETER format, in which TRUE specifies that the value is the true generated value rather than the input specification
(e.g., GEN_Hedge Insertion:Hedge Insertion:TRUEdown_kind_of for the 'kind of' downtoner).
- resources/parameters/models: arff files and binary Weka models for different generation methods (e.g. direct = parameter estimation, seq-pe = sequential parameter estimation, and overgeneration selection models for each trait).
- resources/parameters/data: evaluation data from which the training datasets are extracted.

3.4 Configuration file options

PERSONAGE requires a Java .properties configuration file, such as the PersonageMATCH.properties file in the root directory. The configuration file must specify values for the following options:

- StylisticDimensions: should be set to 'BigFive', as it is the only current set of supported dimensions.
- JWNLProperties: pointer to the JWNL WordNet Java interface property file (e.g. lib/jwnl/file_properties.xml)
- MRCDatabaseDirectory: pointer to the MRC Psycholinguistic database directory (default is resources/lexical/mrc).
- SelectionDatabase: XML database of entities to be compared/recommended (e.g. input/db/restwutils.xml).
- TextPlansDirectory: if predefined text plans are to be used rather than the TextPlanBuilder, this should point to the text plan file directory (e.g. input/tplans).
- DSSDirectory: directory containing the generation dictionary in XML DSyntS format, for every possible text plan proposition (e.g. resources/dsynts/match for DSyntS's for the MATCH restaurant comparison domain, or resources/dsynts/caminfo for the Cambridge Tourist Information (CamInfo) domain).

- HedgeDatabase: XML file containing a database of pragmatic markers in DSyntS format. Each DSyntS must contain an `<instertion-point>` tag that specifies the boundary between (a) the external DSyntS pattern to be matched in the utterance and (b) the marker to be inserted (e.g. `resources/dsynts/markers/markers-templates.xml`).

- DefaultParameterFile: XML file containing parameter values to be used in case no or a neutral style is specified (default is `resources/parameters/handcrafted/params-all-avg.xml`).

- StylisticParametersDir: Directory containing XML parameter files for each end of each dimensions (e.g. one high and low file for each Big Five trait). E.g. `resources/parameters/handcrafted/bigfive`.

- WordNetSenseFile: text file containing sense-tagged word list, based on the `WORD#PART_OF_SPEECH#SENSE_NUMBER` format. To avoid changing the meaning of the utterance, only sense-tagged words can be substituted at run-time (e.g. `resources/lexical/wn_senses/wn_senses.txt`).

- AdjDatabaseFile: list of adjectives split into polarity groups. Adjectives in the original DSyntS can be substituted by any of the adjectives in the polarity group matching the proposition's polarity (e.g. high price = low polarity). E.g. `resources/lexical/adjectives/adj_single_filtered_by_hand.txt`.

- FreqDatabaseFile: list of word frequency counts to be used for the FREQUENCY OF USE lexical selection parameter (e.g. `resources/ lexical/frequencies/bnc_cutoff.txt`).

- VerbOceanDatabase: database from the VERBOCEAN distribution containing stronger-then relation between verbs. This is used by the VERB STRENGTH lexical selection parameter (e.g. `resources/lexical/verbocean/verbocean.unrefined.2004-05-20.txt`).

- OutputSemantics: boolean determining whether the generator outputs aligned semantics if present in the input DSyntS (CamInfo only). Default value is `false`.

3.5 Package structure

PERSONAGE's source code is divided into Java packages with the following functionalities (see `src` directory):

- `generator`: main Personage wrapper classes, e.g. PersonageRuleBased or PersonagePE. The class DSSTransformer applies all transformation rules sequentially. Also contains the RealPro wrapper (Realizer) which post-processes the realisations.

- `generator.aggregation`: SParKy's aggregation functions.
- `generator.aggregation.elements`: XML elements being used as part of a sentence plan tree.
- `generator.aggregation.operations`: aggregation operations combining DSyntS pairs.
- `generator.dssrules`: syntactic transformation rules modifying the Utterance object, e.g. hedge insertion, synonym replacement, pronominalisation, etc.
- `generator.gui`: graphical user interface (main class is ControlFrame).
- `generator.jni`: Java Native Interface classes, to call Personage from C programs.
- `generator.model`: parameter estimation (PE) and overgeneration selection models, e.g. PE models dependent only on the stylistic dimensions (IndependentParameterModel) or dependent on previous parameter values as well (SequentialParameterModel), or parameter estimation by searching over the input space of selection models (RandomSearch).
- `generator.parameters`: contains the data structure holding the generation parameter values at run-time.
- `generator.parameters.dimensions`: defines the set of variation dimensions to be controlled, as well as default parameter files associated with them (e.g. BigFive).
- `generator.textplan`: text plan builder, i.e. produces XML text plans to be sent to the personage generator, e.g. based on a set restaurants to be compared. Contains a TextPlanBuilder for the MATCH and CamInfo domains.

Chapter 4

Generation framework

Our method for generating personality consists of two main components: (1) the PERSONAGE base generator, which produces language expressing various personality traits by implementing the generation decisions described in Chapter 5, and (2) a *personality model* of how the generation decisions influence the projected personality.

The following chapters evaluate three alternative approaches to personality generation, each with a different personality model controlling the base generator. The *rule-based* approach associates each personality type with a set of abstract parameter values derived from psychological studies,[1] and the setting associated with the desired trait is selected at generation time. The parameter files for each end of each Big Five trait are encoded in XML format in the `lib/parameters` directory (the user can manually modify their values through the `value` attributes if needed, although it is not recommended). This personality model can therefore target each end of the Big Five scales. The derivation of the parameter sets is detailed in Mairesse [2008], as well as an evaluation of the rule-based approach Mairesse and Walker [2007]. The second method, referred to as *overgenerate and select*, requires the generation of many candidate utterances using the base generator by randomly varying its input parameters. The personality model is then used to predict the personality score of each utterance, and the utterance yielding the closest score to the target is selected. An evaluation of this method can be found in Mairesse [2008] as well. In the third approach, *parameter estimation models* estimate the optimal generation parameters given target personality scores, which are then used by the base generator to produce the output utterance [Mairesse and Walker, 2008]. The last two approaches model personality variation continuously, whereas the rule-based method only produces extreme personality. The overall methodology behind PERSONAGE can be summarised in the following steps:

1. Developing the base generator:

[1] Parameter values are abstract in the sense that they only indicate a trend as opposed to exact generation decisions, because of the imprecise nature of the psychology findings.

(a) Identify personality markers from psychological studies;

(b) Map these markers to natural language generation decisions;

- **Rule-based generation mode:** derive parameter settings for both ends of each trait from the mappings defined in (b), and use them for generating extreme personality. See Section 6.1.

2. Training the generator's personality model:

(a) Generate utterances covering the full parameter range;

(b) Judges rate the output of step (a) with a standard personality test;

(c) Compute feature values for each utterance based on the actual decisions of the generator and possibly other utterance features;

- **Overgenerate and select mode:** train a statistical model to predict the judges' ratings from the features, and use it to rank randomly generated utterances based on the target personality. See Section 6.2.
- **Parameter estimation mode:** train a statistical model to predict the generation decisions from the judges' ratings, and generate the target personality using the predicted parameter values. See Section 6.3.

The following chapter describe the generator's overall architecture and implementation, while Chapter 6 provides information about how to run the generator in different modes.

Chapter 5

Overview of the base generator

This chapter reviews in detail the generation decisions implemented in the PERSONAGE language generator. At generation time, their values are typically determined by the target personality scores. This chapter provides the necessary information to modify the behaviour of the generator, if needed. The content of this chapter can also be found in Mairesse [2008].

5.1 PERSONAGE's input

PERSONAGE's input consists of a selection of restaurants in New York City, with associated scalar values representing evaluative ratings for six attributes: *food quality*, *service*, *cuisine*, *location*, *price* and *atmosphere*.[1]

The second input consists of parameter values for the generation parameters. As part of the purpose of an independent language generator is to be re-usable in different applications, an objective of the current work is to make PERSONAGE as domain-independent as possible. A consequence is that parameter values are normalised between 0 and 1 for continuous parameters, and to 0 or 1 for binary parameters. For example, a VERBOSITY parameter set to 1 maximises the utterance's verbosity given the input, regardless of the actual number of propositions expressed.

5.2 PERSONAGE's architecture

PERSONAGE implements the traditional pipelined natural language generation (NLG) architecture [Reiter and Dale, 2000], which consists of a series of sequential components, with each component processing the output of its predecessor. The high-level architecture of the base generator is illustrated in Figure 5.1. The first component is the *content planner*,[2] which specifies the structure of the information to be

[1] The attribute values used in the present work are derived from Zagat Survey's ratings, and mapped from a 30-point scale to the $[0, 1]$ interval.

[2] The content planning phase is sometimes referred to as *text planning*.

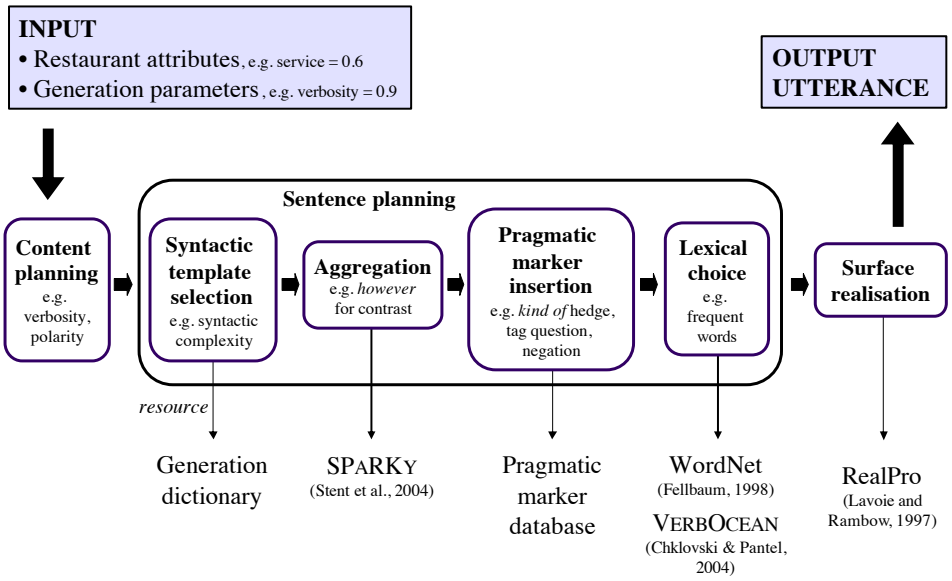


Figure 5.1: The architecture of the PERSONAGE base generator.

conveyed. The resulting content plan tree is then processed by the *sentence planner*,[3] which selects syntactic templates for expressing individual propositions, and aggregates them to produce the utterance's full syntactic structure. The pragmatic marker insertion component then modifies the syntactic structure locally to produce various pragmatic effects, depending on the markers' insertion constraints. The next component selects the most appropriate lexeme for each content word, given the lexical selection parameters. Finally, the RealPro realiser [Lavoie and Rambow, 1997] converts the final syntactic structure into a string by applying surface grammatical rules, such as morphological inflection and function word insertion.

In a typical dialogue system, the output of the realiser is annotated for prosodic information by the prosody assigner, before being sent to the text-to-speech engine to be converted into an acoustic signal. As shown in Figure 5.2, this thesis focuses strictly on linguistic processing and leaves prosody assignment to future work. However, the same methodology could be applied to express personality through prosody as many markers of personality have been identified in speech. The following sections describe each component in more detail.

5.3 Implementation of generation decisions

This section focuses on the implementation of the generation parameters in the PERSONAGE generator. Components are described in data processing order accord-

[3]The sentence planning phase is sometimes referred to as *micro-planning*.

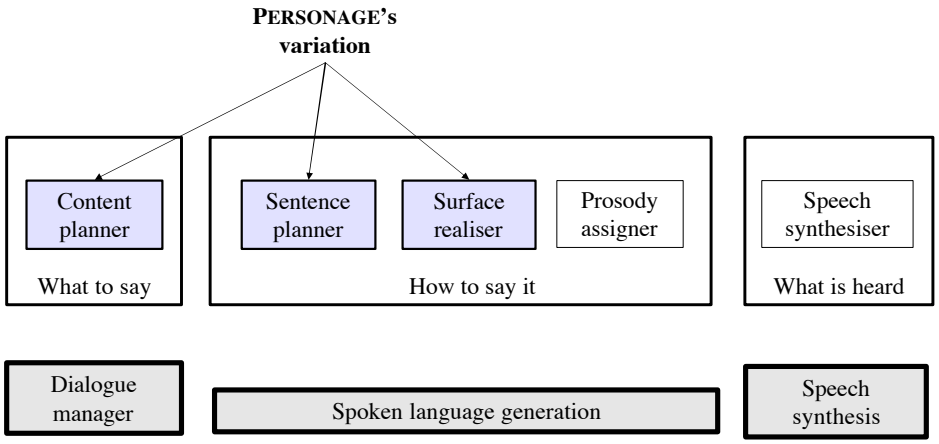


Figure 5.2: The spoken language generation pipelined architecture, with components included in PERSONAGE in the bold inner boxes.

ing to the architecture in Figure 5.1, together with the mechanisms underlying each generation decision.

5.3.1 Content planning

The first step of the generation process is to convert the restaurant's attribute values into a *content plan*, a high level structure reflecting the overall communicative goal of the utterance. In a dialogue system, the initial content plan would be obtained from the dialogue manager. The content plan combines together propositions expressing information about individual attributes using *rhetorical relations* from Mann and Thompson's Rhetorical Structure Theory (RST; 1988). Section **??** presented the structure of PERSONAGE's input content plan tree, which is processed by a number of parameters.

Content size: Extraverts are more talkative than introverts [Furnham, 1990, Pennebaker and King, 1999], although it is not clear whether they actually produce more content, or are just redundant and wordy. Thus various parameters relate to the amount and type of content produced. The VERBOSITY parameter controls the number of propositions selected from the content plan. The parameter value defines the ratio of propositions that are kept in the final content plan tree, while satisfying constraints dependent on the communicative goal: a recommendation must include a claim, and a comparison must include a pair of contrasted propositions. Whereas the VERBOSITY parameter defines the number of propositions in the final content plan, parameters controlling polarity determine what propositions are selected (see below).

The REPETITION parameter adds an exact repetition: the proposition node is duplicated and linked to the original content by a RESTATE rhetorical relation. The continuous parameter value (between 0 and 1) is mapped linearly to the number

of repetitions in the content plan tree, i.e. between 0 and a domain-specific maximum (set to 2 in our domain). Rather than copying the existing proposition, the RESTATEMENT parameter adds a paraphrase to the content plan, obtained from the generation dictionary (see Section 5.3.2). If no paraphrase is found, one is created automatically by substituting content words with the most frequent WordNet synonym (see Section 5.3.5).

Polarity: Extraverts are more positive; introverts are characterised as engaging in more 'problem talk' and expressions of dissatisfaction [Thorne, 1987]. To control for polarity, propositions are defined as positive or negative based on the scalar rating of the corresponding attribute, normalised between 0 and 1. The claim in a recommendation is assigned a maximally positive polarity of 1, whereas the *cuisine* and *location* attributes have a neutral polarity, i.e. a domain-dependent constant set to .58 for our restaurant database.[4] Given a selected restaurant, all other attributes are defined as negative or positive depending on whether their normalised scalar value is below or above the neutral point. There are multiple parameters associated with polarity. The CONTENT POLARITY parameter controls whether the content is mostly negative (e.g. *'Chanpen Thai has mediocre food'*), neutral (e.g. *'Le Marais is a French restaurant'*), or positive (e.g. *'Babbo has fantastic service'*). If there is enough polarised content given the required content plan tree size, the following propositions are selected depending on the input CONTENT POLARITY parameter:

CONTENT POLARITY	$\in [0, .25[$	only negative propositions
	$\in [.25, .5[$	negative and neutral propositions
	$\in [.5, .75[$	neutral and positive propositions
	$\in [.75, 1]$	only positive propositions

If there are not enough propositions in the resulting set to satisfy the verbosity constraint, propositions with the closest polarity are added until the required content plan size is reached. Additionally, a constraint requiring that a comparison content plan tree contains at least one CONTRAST relation is enforced, thus the tree is likely to include propositions with different polarities.

From the filtered set of propositions, the POLARISATION parameter determines whether the final content includes attributes with extreme scalar values or not (e.g. *'Chanpen Thai has fantastic staff'* vs. *'Chanpen Thai has decent staff'*). The final content plan tree therefore contains the propositions whose normalised distance to the neutral point is the closest to the target POLARISATION value ($\in [0, 1]$), while its size is defined by the VERBOSITY constraint.

In addition, polarity can also be implied more subtly through rhetorical structure. The CONCESSIONS parameter controls the way in which negative and positive information is presented, i.e. whether two propositions with different polarity are

[4] This neutral value was chosen based on the perception of Zagat Survey's restaurant ratings, i.e. scores below $\frac{17.5}{30} = .58$ are considered negative.

presented objectively, or if one is foregrounded and the other backgrounded. If two opposed propositions are selected for a concession, a CONCEDE mononuclear rhetorical relation is inserted between them. More precisely, the parameter controls the ratio of concessions being inserted out of all proposition pairs with opposite polarity.[5] While the CONCESSIONS parameter captures the tendency to put information into perspective, the CONCESSION POLARITY parameter controls whether the positive or the negative content is conceded, i.e. marked as the satellite of the CONCEDE relation (e.g. *'even if the food is good, it's expensive'* vs. *'even if the food is expensive, it's good'*). The parameter value determines the ratio of positive concessions out of the total number of concessions in the content plan tree.

Content ordering: Although extraverts use more positive language [Thorne, 1987, Pennebaker and King, 1999], it is unclear how they position the positive content within their utterances. Additionally, the position of the claim affects the persuasiveness of an argument [Carenini and Moore, 2000]. The POSITIVE CONTENT FIRST parameter therefore controls whether positive propositions—including the claim—appear first or last, i.e. the order in which the propositions are aggregated. The parameter controls the ratio of sibling proposition pairs that are ordered with increasing polarity. Although this parameter determines the ordering of the nodes of content plan tree, some aggregation operations can still impose a specific ordering (e.g. BECAUSE CUE WORD to realise the JUSTIFY relation, see Section 5.3.3).

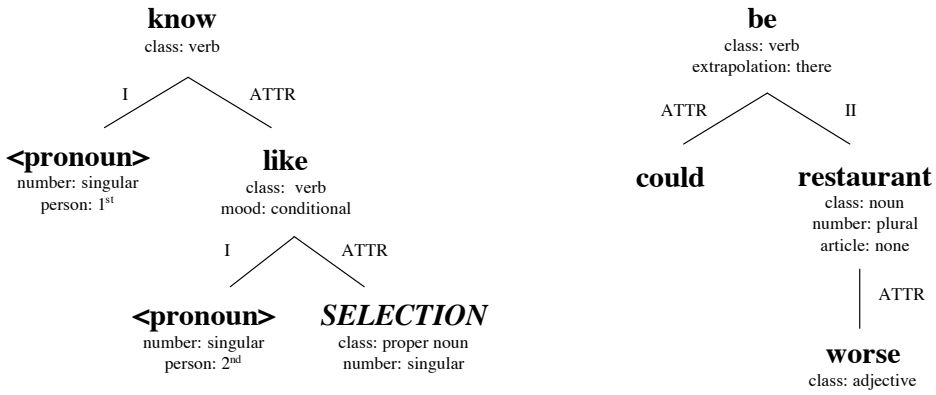
While the INITIAL REJECTION, REQUEST CONFIRMATION and COMPETENCE MITIGATION parameters can also be seen as content planning parameters, they are modelled at the pragmatic marker insertion level as they only affect the beginning of the utterance (see Section 5.3.4).

5.3.2 Syntactic template selection

Once the content planner has determined *what* will be talked about in the utterance, the remaining components control *how* the information is to be conveyed. The first sentence planning component associates each proposition in the content plan with a syntactic template. PERSONAGE manipulates syntactic templates referred to as Deep Syntactic Structures (DSyntS). DSyntS are syntactic representations inspired by Melčuk's Meaning-Text Theory [1988], a linguistic framework in which language is modelled as a multi-stage rule-based process, that gradually modifies the utterance representation from semantics to text. DSyntS can be converted to a text string using the RealPro surface realiser [Lavoie and Rambow, 1997]. The templates are stored in a small handcrafted generation dictionary, containing 18 DSyntS: 12 for the recommendation claim and one per attribute. The DSyntS can contain variables that are filled at generation time, such as the restau-

[5] We only consider concessions between attributes of the same restaurant.

rant's name or cuisine. Figure 5.3 shows two DSyntS expressing the recommendation claim. The DSyntS selection process assigns each candidate DSyntS to a point in a three-dimensional space, characterising the DSyntS' syntactic complexity, number of self-references and polarity. Parameter values are normalised over all candidate DSyntS, so the DSyntS closest to the target values can be computed.



(a) *I know you would like SELECTION*

(b) *There could be worse restaurants*

Figure 5.3: Two example DSyntS for a recommendation claim. The lexemes are in bold, and the attributes below indicate non-default values in the RealPro realiser. Branch labels indicate dependency relations, i.e. I = subject, II = object and ATTR = modifier. Lexemes in italic are variables that are instantiated at generation time.

Syntactic complexity: Furnham [1990] suggests that introverts produce more complex constructions: the SYNTACTIC COMPLEXITY parameter controls the number of subordinate clauses of the DSyntS chosen to represent the claim, based on Beaman's definition of syntactic complexity [1984].[6] For example, the claim in Figure 5.3(a) is rated as more complex than the one in Figure 5.3(b), because the latter has no subordinate clause.

Self-references: As extraverts and neurotics make more self-references [Pennebaker and King, 1999], the SELF-REFERENCES parameter controls whether the claim is made in the first person, based on the speaker's own experience, or whether the claim is reported as objective or information obtained elsewhere. The SELF-REFERENCES value is computed from the DSyntS by counting the number of first person pronouns. For example, the template in Figure 5.3(a) contains one self-reference, while the template in Figure 5.3(b) does not.

[6]The syntactic complexity is computed as the number of verb nodes in the DSyntS, which is equivalent to the number of subordinate clauses in the final utterance.

Polarity: While polarity can be expressed by content selection and structure, it can also be directly associated with the DSyntS. The TEMPLATE POLARITY parameter determines whether the claim has a positive or negative connotation. An example claim with low polarity can be found in Figure 5.3(b), i.e. *'There could be worse restaurants'*, whereas the claim in Figure 5.3(a) is rated more positively. The templates used in PERSONAGE can be found in the `dss/dictionary` directory, as DSyntS in XML format. Each file represents a generation dictionary for a specific dialogue act, some of which are annotated for polarity using the `polarity` attribute of the `dsynts` root node (e.g. file `assert-reco-best.xml`). Polarity is expressed as a real value ranging from 0 (very negative) to 1 (very positive).

5.3.3 Aggregation

RST relation	Aggregation operations
JUSTIFY	WITH CUE WORD, RELATIVE CLAUSE, SO CUE WORD, BECAUSE CUE WORD, SINCE CUE WORD, PERIOD
CONTRAST	MERGE, HOWEVER CUE WORD, WHILE CUE WORD, CONJUNCTION, BUT CUE WORD, ON THE OTHER HAND CUE WORD, PERIOD
INFER	MERGE, WITH CUE WORD, RELATIVE CLAUSE, ALSO CUE WORD, CONJUNCTION, PERIOD
CONCEDE	EVEN IF CUE WORD, ALTHOUGH CUE WORD, BUT/THOUGH CUE WORD
RESTATE	CONJUNCTION, MERGE WITH COMMA, OBJECT ELLIPSIS

Table 5.1: Clause combining operations for different rhetorical relations, based on SPARKY's operations [Stent et al., 2004, Walker et al., 2007].

The role of the aggregation component is to combine syntactic templates together into a larger syntactic structure, by associating each pair of sibling propositions in the content plan tree with a *clause-combining operation* that determines how the parent rhetorical relation is to be expressed. For example, poor food quality can be contrasted with good atmosphere using cue words such as 'however' or 'but'. For each rhetorical relation in the content plan tree, the aggregation process randomly selects a clause-combining operation according to the probability distribution for that relation defined by the input aggregation parameters, e.g. the distributions for the INFER relation in Figure 5.3. The aggregation process then selects pairs of operation arguments among the children propositions, until the two associated DSyntS satisfy the constraints of the clause-combining operation, e.g. the MERGE operation requires that both argument DSyntS have the same main verb. If none of the pairs satisfy the constraints, another clause-combining operation is chosen according to the input probability distribution. The aggregation process is guaranteed to terminate as each rhetorical relation implements at least one clause-combining operation with no constraint on the DSyntS, i.e. the PERIOD operation, which keeps both argument DSyntS in separate sentences. PERSONAGE uses the SPARKY clause-combining operations [Stent et al., 2004], with additional operations for the RESTATE and CONCEDE rhetorical relations. Table 5.1 shows some

Operation	Relations	Description	Sample 1st arg	Sample 2nd arg	Result
MERGE	INFER or CONTRAST	Two clauses can be combined if they have identical verbs and identical arguments and adjuncts except one. The non-identical arguments are coordinated.	Chanpen Thai has good service.	Chanpen Thai has good food quality.	Chanpen Thai has good service and good food quality.
WITH CUE WORD	JUSTIFY or INFER	Two clauses with identical subject arguments can be identified if one of the clauses contains the verb *to have*. The possession clause undergoes *with*-participial clause formation and is attached to the non-reduced clause.	Chanpen Thai is a Thai restaurant.	Chanpen Thai has good food quality.	Chanpen Thai is a Thai restaurant, with good food quality.
RELATIVE CLAUSE	JUSTIFY or INFER	Two clauses with an identical subject can be identified. One clause is attached to the subject of the other clause as a relative clause.	Chanpen Thai has good food quality.	Chanpen Thai is located in Midtown West.	Chanpen Thai, which is located in Midtown West, has good food quality.
CONJUNCTION	JUSTIFY, INFER or CONTRAST	Two clauses are conjoined with a coordinating conjunction. They are separated by a comma if the right clause already contains a conjunction.	Chanpen Thai has good food quality.	Chanpen Thai has good service.	Chanpen Thai has good food quality and it has good service.
ON THE OTHER HAND CUE WORD	CONTRAST	Combines clauses by inserting a cue word at the start of the second clause, resulting in two separate sentences.	Chanpen Thai has very good decor.	Baluchi's has mediocre decor.	Chanpen Thai has very good decor. On the other hand, Baluchi's has mediocre decor.
EVEN IF CUE WORD	CONCEDE	Combines clauses by inserting the *even if* adverbial at the start of the satellite clause. The order of the arguments is determined by the order of the nucleus (N) and the satellite (S), yielding two distinct operations, EVEN IF CUE WORD NS and EVEN IF CUE WORD SN.	Chanpen Thai has very good decor.	Chanpen Thai's has mediocre food quality.	Chanpen Thai has very good decor, even if it has mediocre food quality.
MERGE WITH COMMA	RESTATE	Merges repeated clauses in the same way as the MERGE operation, but ensures that the non-identical arguments are separated by a comma.	Chanpen Thai has very good service.	Chanpen Thai has fantastic waiters.	Chanpen Thai has very good service, fantastic waiters.
OBJECT ELLIPSIS	RESTATE	Coordinates clauses and replaces the object of the first clause by a three-dot ellipsis.	Chanpen Thai has very good service.	Chanpen Thai has fantastic waiters.	Chanpen Thai has... It has fantastic waiters.
PERIOD	Any	Two clauses are joined by a period.	Chanpen Thai is a Thai restaurant, with good food quality.	Chanpen Thai has good service.	Chanpen Thai is a Thai restaurant, with good food quality. It has good service.

Table 5.2: Clause-combining operations and examples as described in previous work on the SPARKY generator [Stent et al., 2004, Walker et al., 2007], together with new operations specific to PERSONAGE.

of the available operations for each rhetorical relation; their effect on the final utterance is illustrated in Table 5.2.

Aggregation parameters for the INFER relation	Introvert distribution	Extravert distribution
INFER - MERGE	.20	.50
INFER - RELATIVE CLAUSE	.40	.00
INFER - WITH CUE WORD	.30	.10
INFER - ALSO CUE WORD	.00	.10
INFER - CONJUNCTION	.00	.29
INFER - PERIOD	.10	.01

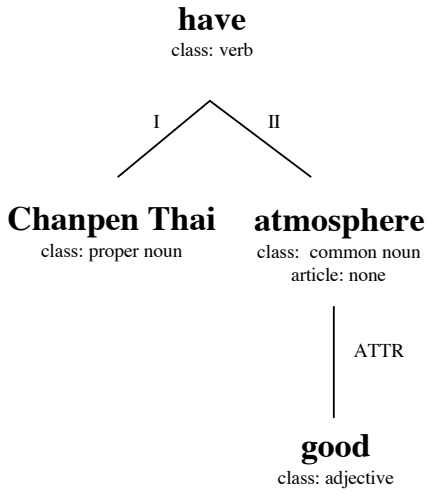
Table 5.3: Probability distribution of aggregation operations expressing the INFER relation for the introvert and extravert parameter settings.[7] Parameter values must add up to 1.

The probability of the operations biases the production of complex clauses, full stops and formal cue words for introverts, to express their preference for complex syntactic constructions, long pauses and rich vocabulary [Furnham, 1990]. Thus, the introvert parameters favour operations such as RELATIVE CLAUSE and PERIOD for the INFER relation, HOWEVER CUE WORD for CONTRAST, and ALTHOUGH CUE WORD for CONCEDE, that we hypothesise to result in more formal language. Extravert aggregation produces longer sentences with simpler constructions and informal cue words. Thus extravert utterances tend to use operations such as a CONJUNCTION to realise the INFER and RESTATE relations, and the EVEN IF CUE WORD for CONCEDE relations. Aggregation parameter values for expressing the INFER relation are illustrated in Table 5.3, for both introvert and extravert parameter settings.

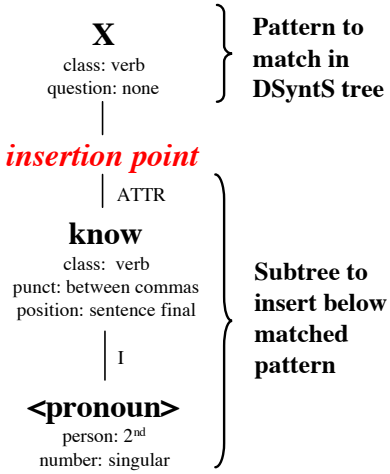
5.3.4 Pragmatic marker insertion

Many personality markers are not related to content selection or structuring, rather they manifest themselves through localised syntactic elements reflecting pragmatic effects that only affect a small part of the utterance. To control the insertion of such markers, PERSONAGE implements a pragmatic marker insertion component. A handcrafted database contains syntactic elements characterising each pragmatic marker. For each marker, the insertion process involves traversing the aggregated DSyntS to identify *insertion points* satisfying the syntactic constraints specified in the database. Figure 5.4 illustrates the matching and insertion process for the hedge *you know*. Each entry in the marker database consists of a syntactic pattern to be matched in the DSyntS, such as the root node in Figure 5.4(b), and an insertion point element corresponding to the location in the DSyntS where to insert the

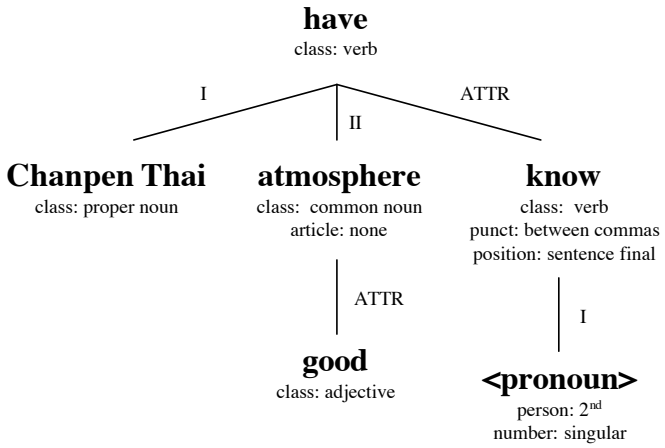
[7] The input selection probability does not entirely reflect the probability that an operation will appear in the output utterance, as the latter is also dependent on the constraints the operation imposes on its DSyntS arguments. For example, the MERGE operation requires both DSyntS to have the same verb, while the CONJUNCTION operation does not. Thus, individual probabilities are scaled to counterbalance these constraints.



(a) Example input DSyntS realised as *'Chanpen Thai has good atmosphere'*.



(b) Syntactic representation of the insertion constraints for the pragmatic marker *you know*.



(c) Modified DSyntS after the insertion of the pragmatic marker below the main verb matching the pattern defined in Figure 5.4(b)'s root node.

Figure 5.4: Illustration of the pragmatic marker insertion process for the hedge *you know* in the DSyntS *'Chanpen Thai has good atmosphere'*.

subtree representing the marker. Given the input DSyntS in Figure 5.4(a) *'Chanpen Thai has good atmosphere'*, the verb *to have* is matched with the root node of the structure in Figure 5.4(b), and thus the subtree below the insertion point is inserted under Figure 5.4(a)'s root node. The resulting DSyntS is in Figure 5.4(c),

realised as *'Chanpen Thai has good atmosphere, you know'*. The utterance is modified at the syntactic level rather than at the surface level, to reduce the complexity of each operation by relying on the surface realiser for grammaticality. For example, pragmatic markers are added without controlling the final word order, while positional constraints can be enforced when required, e.g. the *position* attribute in Figure 5.4(b) specifies that *you know* should be in sentence final position. Similarly, while the *punct* attribute specifies that the marker must appear between commas—irrespectively of its position in the utterance, the realiser ensures that the sentence is punctuated correctly by removing commas preceding the final full stop.

Syntactically embedded markers: PERSONAGE implements a binary generation parameter for most pragmatic markers listed in Tables 5.4 and 5.5, using the insertion mechanism detailed in the previous paragraph. At generation time, syntactic patterns are randomly chosen (with a uniform distribution) among markers with parameter values set to 1, and matched against the aggregated DSyntS. The insertion process ends when there are no markers left in the database, or when the number of successful insertions is above a constant threshold (heuristically set to 5 for the current domain) to avoid producing unnatural utterances. Each marker and their insertion patterns are stored in XML format in the file `dss/markers/markers-templates.xml`, as a list of partial DSyntS containing an `insertion-point` node defining the boundary between the insertion pattern and the actual pragmatic marker. The pragmatic marker database file can be modified, however great caution must be taken as it could result in ungrammatical outputs.

Other markers: While most pragmatic markers are implemented as described above, additional markers require more complex syntactic processing and are implemented independently.

Referring expression generation is a traditional problem in NLG [Reiter and Dale, 2000], which is solved in PERSONAGE by pronominalising any occurrence of a restaurant name following a reference to the same selection, e.g. *'Chanpen Thai is the best, it has great service'*. However, proximal deictic expressions can be seen as a way to express involvement and empathy [Brown and Levinson, 1987], e.g. *'this restaurant has great service'*. Thus, a PRONOMINALISATION parameter controls whether referring expressions are expressed as personal pronouns or proximal demonstrative phrases, by specifying the ratio of pronouns out of all referring expressions in the utterance. Concerning the implementation, the RealPro surface realiser automatically selects the personal pronoun based on the selection's DSyntS node; inserting a demonstrative phrase requires replacing the selection's lexeme with a generic noun (e.g. *restaurant*) and setting the determiner to a demonstrative.

As negations indicate both introversion and a lack of conscientiousness [Pennebaker and King, 1999, Mehl et al., 2006], a NEGATION parameter inserts a negation while preserving the initial communicative goal. If the parameter is enabled,

Marker	Constraints	Example
General markers:		
NEGATION*	adjective modifier with antonym	Chanpen Thai doesn't have bad atmosphere
EXCLAMATION	sentence-final punctuation	Chanpen Thai has good atmosphere!
IN-GROUP MARKER	clause-final adjunct; available markers are *pal*, *mate* and *buddy*	Chanpen Thai has good atmosphere pal
SUBJECT IMPLICITNESS*	requires a DSyntS of the form *NOUN has ADJ NOUN*	The atmosphere is good
TAG QUESTION*	none	Chanpen Thai has good atmosphere, doesn't it?
STUTTERING*	selection name	Ch-Chanpen Thai has good atmosphere
EXPLETIVES	adjective modifier (*damn*, *bloody*)	Chanpen Thai has damn good atmosphere
	clause-initial adjunct (*oh god*)	Oh god Chanpen Thai has good atmosphere
NEAR EXPLETIVES	adjective modifier (*darn*)	Chanpen Thai has darn good atmosphere
	clause-initial adjunct (*oh gosh*)	Oh gosh Chanpen Thai has good atmosphere
REQUEST CONFIRMATION*	none	You want to know more about Chanpen Thai? Let's see... Chanpen Thai Let's see what we can find on Chanpen Thai Did you say Chanpen Thai?
INITIAL REJECTION*	none	I don't know I'm not sure I might be wrong
COMPETENCE MITIGATION	main verb is subordinated to new clause (*everybody knows that* and *I thought everybody knew that*)	Everybody knows that Chanpen Thai has good atmosphere
	clause-initial adjunct (*come on*)	Come on, Chanpen Thai has good atmosphere
Softeners:		
KIND OF	adjective modifier	Chanpen Thai has kind of good atmosphere
SORT OF	adjective modifier	Chanpen Thai has sort of good atmosphere
SOMEWHAT	adjective modifier with verb *to be*	The atmosphere is somewhat good
QUITE	adjective modifier	Chanpen Thai has quite good atmosphere
RATHER	adjective modifier	Chanpen Thai has rather good atmosphere
AROUND	numeral modifier	Chanpen Thai's price is around $44
SUBORDINATE	main verb is subordinated to new clause; available clauses are *I think that* and *it seems (to me) that*	It seems to me that Chanpen Thai has good atmosphere
Filled pauses:		
LIKE	verb modifier	Chanpen Thai has, like, good atmosphere
ERR	clause-initial adjunct	Err... Chanpen Thai has good atmosphere
MMHM	clause-initial adjunct	Mmhm... Chanpen Thai has good atmosphere
I MEAN	clause-initial adjunct	I mean, Chanpen Thai has good atmosphere
YOU KNOW	clause-final adjunct	Chanpen Thai has good atmosphere, you know

Table 5.4: Pragmatic markers implemented in PERSONAGE, with insertion constraints and example realisations. An asterisk indicates that the pragmatic marker requires specific processing and was not implemented through pattern matching and insertion.

Marker	Constraints	Example
Emphasisers:		
REALLY	adjective modifier	Chanpen Thai has really good atmosphere
BASICALLY	clause-initial adjunct	Basically, Chanpen Thai has good atmosphere
ACTUALLY	clause-initial adjunct	Actually, Chanpen Thai has good atmosphere
JUST	pre-verbal modifier of *to have*	Chanpen Thai just has good atmosphere
	post-verbal modifier of *to be*	The atmosphere is just good
Acknowledgment markers:		
YEAH	clause-initial adjunct	Yeah, Chanpen Thai has good atmosphere
WELL	clause-initial adjunct	Well, Chanpen Thai has good atmosphere
OH	clause-initial adjunct	Oh, Chanpen Thai has good atmosphere
RIGHT	clause-initial adjunct	Right, Chanpen Thai has good atmosphere
OK	clause-initial adjunct	Ok, Chanpen Thai has good atmosphere
I SEE	clause-initial adjunct	I see, Chanpen Thai has good atmosphere

Table 5.5: Pragmatic markers implemented in PERSONAGE (second part), with insertion constraints and example realisations. An asterisk indicates that the pragmatic marker requires specific processing and was not implemented through pattern matching and insertion.

an adjective modifying a verb or its object is randomly selected from the DSyntS, and its antonym is retrieved from WordNet [Fellbaum, 1998]. If the query is successful, the adjective's lexeme is replaced by the antonym and the governing verb is negated,[8] e.g. *'Chanpen Thai has good atmosphere'* becomes *'Chanpen Thai doesn't have bad atmosphere'*. Adjectives in the domain are manually sense-tagged to ensure that they can be substituted by their antonym. Also, a maximum of one negation can be inserted to prevent the utterance from sounding unnatural.

Heylighen and Dewaele [2002] found that extraverts use more implicit language than introverts. A SUBJECT IMPLICITNESS parameter thus determines whether predicates describing restaurant attributes are expressed with the restaurant's name in the subject, or with the attribute itself by making the reference to the restaurant implicit (e.g. *'Chanpen Thai has good atmosphere'* vs. *'the atmosphere is good'*). The syntactic transformation involves shifting the object attribute to the subject, while promoting the adjective below the main verb, and changing the main verb's lexeme to *to be*. Hence, the transformation requires an input DSyntS matching the template *NOUN has ADJECTIVE NOUN*.

As speech disfluencies are associated with anxiety and neuroticism [Scherer, 1981], a STUTTERING parameter modifies the lexeme of a randomly selected proper noun by repeating the first two letters two or three times, e.g. *'Ch-Ch-Chanpen Thai'*. Only selection names are repeated as they are likely to be new to the speaker, the stuttering can therefore be interpreted as non-pathological. Also, allowing disfluencies to affect any word requires determining what words can be altered, which

[8] At the DSyntS level the negation is represented as an attribute of the verb element, the actual inflection is done by RealPro in the realisation phase.

involves deep psycholinguistic modelling that is beyond the scope of this work.

PERSONAGE also implements politeness markers such as rhetorical questions. The TAG QUESTION parameter processes the DSyntS by (1) duplicating a randomly selected verb and its subject; (2) negating the verb; (3) pronominalising the subject; (4) setting the verb to the interrogative form and (5) appending the duplicated subtree as a sentence-final adjunct, e.g. *'Chanpen Thai has great food'* results in the insertion of *'doesn't it?'*. The duplicated verb is generally not realised,[9] i.e. only the negated auxiliary appears in the tag question. Additionally, whenever the subject is a first person pronoun, the verb is set to the conditional form and a second person pronoun is inserted, producing *'I would recommend Chanpen Thai, wouldn't you?'*. If the tag question insertion is unsuccessful, e.g. due to an extrapolated subject *'there is'*, a default tag question is appended, producing either *'you see?'*, *'alright?'* or *'okay?'*.

The remaining parameters are content level parameters that we consider as pragmatic markers, as they only affect the beginning of the utterance. The first two parameters are implemented by inserting a full DSyntS before the utterance, randomly chosen from a predefined list with a uniform probability.[10] First, the INITIAL REJECTION parameter reduces the level of confidence of the speaker over the utterance's informational content, by beginning the utterance with either *'I don't know'*, *'I'm not sure'* or *'I might be wrong'*. Second, the REQUEST CONFIRMATION parameter produces an implicit confirmation, which both redresses the hearer's positive face through grounding and emphasises the system's uncertainty about the user's request, e.g. *'you want to know more about Chanpen Thai?'*. In order to convey disagreeableness, a COMPETENCE MITIGATION parameter also presents the user's request as trivial by embedding it as a subordinate clause, e.g. *'everybody knows that Chanpen Thai has good service'*. See Table 5.4 for additional example confirmation and competence mitigation DSyntS.

Once PERSONAGE has attempted to insert the pragmatic markers specified by the input parameter setting, the next component selects the final lexical items in the DSyntS.

5.3.5 Lexical choice

PERSONAGE allows many different lexemes to be expressed for each content word, depending on input parameter values.

The lexical selection component processes the DSyntS by sequentially modifying each content word. For each lexeme in the DSyntS, the corresponding WordNet synonyms are mapped to a multi-dimensional space defined by the lexeme's length, frequency of use and strength, using machine-readable dictionaries. The

[9] The verb *to be* is an exception.

[10] The constraint on the maximum number of pragmatic markers in the utterance also affects the insertion probability of the DSyntS.

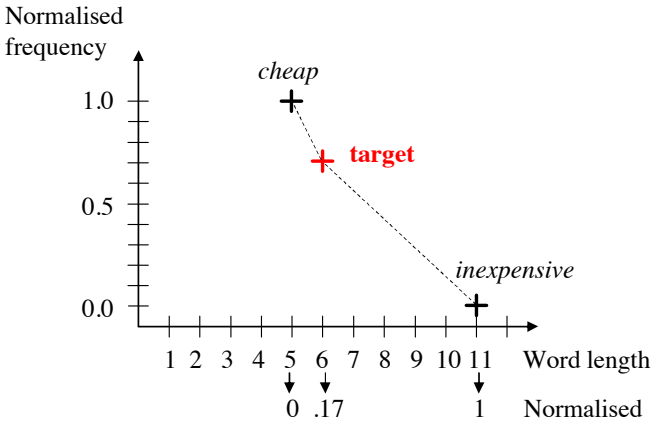


Figure 5.5: Illustration of the lexical selection process between the synonyms *cheap* and *inexpensive* with two input dimensions.

values along each dimensions are normalised over the set of synonyms, and the synonym that is the closest to the target parameter values (in terms of Euclidean distance) is selected. Although word-sense disambiguation techniques could be used in the future, content words are manually sense-tagged to ensure that the synonyms are interchangeable in the dialogue domain. Figure 5.5 illustrates the lexical choice process using the word length and word frequency dimensions, resulting in the selection of *cheap* over *inexpensive* because its length (5 letters) and its normalised frequency (1.0) are closer to the desired target values, i.e. a 6 letter word (normalised length of $\frac{6-5}{11-5} = .17$) with a normalised frequency of .7.

In order to enrich the initial handcrafted pool of synonyms, adjectives extracted by Higashinaka et al. [2007] from a corpus of restaurant reviews and their synonyms are added to the synonym set of each attribute modifier. The list of adjectives is manually filtered for noise. As Higashinaka et al.'s method automatically extracts polarity values for each adjective on a scale from 1 to 5 based on the ratings of the associated reviews, the synonym set for a specific attribute is determined at generation time by mapping the attribute's scalar rating to the polarity scale, e.g. a DSyntS expressing a food quality rating of .42 is mapped to the adjective set with polarity 2 (as $\frac{2}{5} \sim .42$), consisting of the modifiers *bland*, *mediocre* and *bad*. Table 5.6 lists the extracted adjective sets for the food quality attribute, ordered by polarity. The adjectives and their polarities are stored in the file `lib/adj_single_filtered_by_hand.txt`, which can be modified if needed.

The synonym selection is implemented in PERSONAGE by jointly controlling the average normalised frequency of use, word length and verb strength in each DSyntS.

Frequency of use: Introvert and emotionally stable speakers use a richer vo-

Polarity	Adjectives
1	awful, bad, terrible, horrible, horrendous
2	bland, mediocre, bad
3	decent, acceptable, adequate, satisfying
4	good, flavourful, tasty, nice
5	excellent, delicious, great, exquisite, wonderful, legendary, superb, terrific, fantastic, outstanding, incredible, delectable, fabulous, tremendous, awesome, delightful, marvellous

Table 5.6: Adjectives and polarity ratings (5=very positive) for the food quality attribute, extracted from a corpus of restaurant reviews by Higashinaka et al. [2007].

cabulary [Dewaele and Furnham, 1999, Gill and Oberlander, 2003], thus a LEXICON FREQUENCY parameter selects lexical items by their frequency count in the British National Corpus.[11]

Word length: As Mehl et al. [2006] show that observers associate long words with agreeableness, conscientiousness and openness to experience, the LEXICON WORD LENGTH parameter controls the number of letters of the selected synonym.

Verb strength: Verb synonyms differ in terms of their connotative strength, such as *appreciate*, *like* and *love*. This variation is controlled in PERSONAGE through the VERB STRENGTH parameter, which orders each verb's synonym set according to the *stronger-than* semantic relation in the VERBOCEAN database [Chklovski and Pantel, 2004]. The process is illustrated in Figure 5.6 for synonyms of the verb *to know*. The ordered synonyms are mapped to equidistant points in the $[0, 1]$ interval to produce the final parameter value, i.e. the weakest verb is associated with 0.0 and the strongest with 1.0. This mapping is based on the assumption that the magnitude of the *stronger-than* relation is constant between contiguous synonyms, i.e. the verb strength is uniformly distributed over the synonym set. The VERBOCEAN databased is stored in the file `lib/verbocean.unrefined.2004-05-20.txt`.

The lexical choice parameters described above associate each candidate synonym with three values, and the one with the closest values to the target is selected. Since values are normalised over the members of the synonym set, all dimensions have the same weight in the selection process.[12] Consider the input DSyntS expressing *'I know you would like Chanpen Thai'*, a low VERB STRENGTH parameter value produces *'I guess you would like Chanpen Thai'*, whereas a high value yields *'I know you would love Chanpen Thai'*. Similarly, a proposition realised as *'this place has great ambiance'* is converted into *'this restaurant features fantastic atmosphere'* given high LEXICON WORD LENGTH and VERB STRENGTH parameter values together with a low LEXICON FREQUENCY value.

[11] Frequency counts are part-of-speech dependent.

[12] An exception is that verb selection is only affected by the VERB STRENGTH parameter, to ensure that its effect is perceptible in the output utterance.

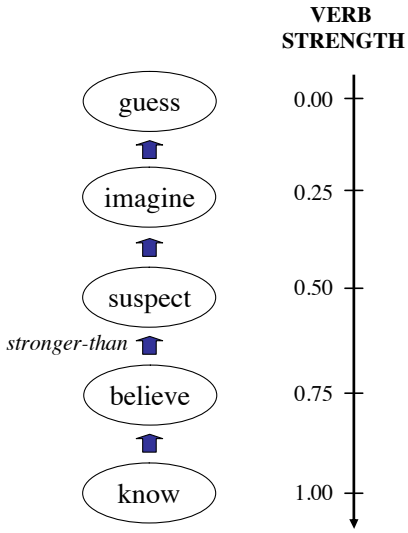


Figure 5.6: Determination of the VERB STRENGTH parameter values for synonyms of the verb *to know*, based on the *stronger-than* semantic relation in VERBOCEAN.

Chapter 6

Generation methods

6.1 PERSONAGE-RB**: Rule-based generation**

6.2 PERSONAGE-OS**: Overgenerate and select**

6.3 PERSONAGE-PE**: Parameter estimation models**

Chapter 7

Tutorial

Bibliography

K. Beaman. Coordination and subordination revisited: Syntactic complexity in spoken and written narrative discourse. In D. Tannen and R. Freedle, editors, *Coherence in Spoken and Written Discourse*, pages 45–80. Ablex, 1984.

P. Brown and S. Levinson. *Politeness: Some universals in language usage*. Cambridge University Press, 1987.

G. Carenini and J. D. Moore. A strategy for generating evaluative arguments. In *Proceedings of International Conference on Natural Language Generation*, pages 47–54, Mitzpe Ramon, Israel, 2000.

T. Chklovski and P. Pantel. VERBOCEAN: Mining the web for fine-grained semantic verb relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, 2004.

J.-M. Dewaele and A. Furnham. Extraversion: the unloved variable in applied linguistic research. *Language Learning*, 49(3):509–544, 1999.

C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

A. Furnham. Language and personality. In H. Giles and W. Robinson, editors, *Handbook of Language and Social Psychology*. Winley, 1990.

A. Gill and J. Oberlander. Perception of e-mail personality at zero-acquaintance: Extraversion takes care of itself; neuroticism is a worry. In *Proceedings of the 25th Annual Conference of the Cognitive Science Society*, pages 456–461, 2003.

F. Heylighen and J.-M. Dewaele. Variation in the contextuality of language: an empirical measure. *Context in Context, Special issue of Foundations of Science*, 7 (3):293–340, 2002.

R. Higashinaka, M. A. Walker, and R. Prasad. An unsupervised method for learning generation lexicons for spoken dialogue systems by mining user reviews. *ACM Transactions on Speech and Language Processing*, 4(4), 2007.

B. Lavoie and O. Rambow. A fast and portable realizer for text generation systems. In *Proceedings of the 3rd Conference on Applied Natural Language Processing*, pages 265–268, 1997.

F. Mairesse. *Learning to Adapt in Dialogue Systems: Data-driven Models for Personality Recognition and Generation*. PhD thesis, University of Sheffield, Department of Computer Science, February 2008.

F. Mairesse and M. A. Walker. PERSONAGE: Personality generation for dialogue. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 496–503, 2007.

F. Mairesse and M. A. Walker. Trainable generation of Big-Five personality styles through data-driven parameter estimation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2008.

W. C. Mann and S. A. Thompson. Rhetorical structure theory. Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

M. R. Mehl, S. D. Gosling, and J. W. Pennebaker. Personality in its natural habitat: Manifestations and implicit folk theories of personality in daily life. *Journal of Personality and Social Psychology*, 90:862–877, 2006.

I. A. Melčuk. *Dependency Syntax: Theory and Practice*. SUNY, Albany, New York, 1988.

J. W. Pennebaker and L. A. King. Linguistic styles: Language use as an individual difference. *Journal of Personality and Social Psychology*, 77:1296–1312, 1999.

E. Reiter and R. Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

K. R. Scherer. Vocal indicators of stress. In J. Darby, editor, *Speech evaluation in psychiatry*, pages 171–187. Grune & Stratton, New York, 1981.

A. Stent, R. Prasad, and M. A. Walker. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2004.

A. Thorne. The press of personality: A study of conversations between introverts and extraverts. *Journal of Personality and Social Psychology*, 53:718–726, 1987.

M. A. Walker, A. Stent, F. Mairesse, and R. Prasad. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research (JAIR)*, 30:413–456, 2007.