



**Universitat
Autònoma
de Barcelona**

MASTER IN COMPUTER VISION AND ARTIFICIAL INTELLIGENCE

REPORT OF THE MASTER PROJECT

OPTION: ARTIFICIAL INTELLIGENCE

Text Modification Methods for Natural Language Generation

Author: **Josep Valls Vargas**

Advisor: **Santiago Ontañón**

Villar

Acknowledgements

We would like to thank the individuals that helped us validate the results of our experiments by providing us with human generated solutions to benchmark our machine generated data. We would also like to thank the communities behind the open source and open knowledge initiatives that helped research design and develop the foundational ideas of this thesis.

ABSTRACT

Natural text generation has been one of the longstanding problems in artificial intelligence. In this thesis, we have focused in the related problem of automatic text modification, which consists on: given a source sentence and a desired change, construct a valid natural language sentence that implements the desired change from the source sentence. This problem is relevant in many template-based text generation systems such as narrative systems. These systems often generate stories tailored to specific users by adapting or instantiating a pre-authored story. Through the thesis we worked with Riu, an analogy-based interactive narrative system. The results show significant improvements for simple sentences in an unconstrained domain.

Keywords: *Natural language, natural text generation, statistical methods, linguistics, template, case-based reasoning, NLP, NLG, CBR, N-gram, Markov chains*

Contents

1	Introduction	1
2	Background Information	3
2.1	Definitions and Concepts	3
2.1.1	Language, Natural Language and Computational Linguistics	3
2.1.2	Sentences, Phrases and Words	4
2.1.3	Language Modeling	4
2.1.4	Linguistic Realization	5
2.2	State of the Art	5
3	Problem Statement	8
3.1	Conceptual Model	11
3.1.1	Case-Based Reasoning for Natural Language Generation	13
3.1.2	Search-Based methods for Natural Language Generation	15
3.1.3	Probabilistic Evaluation of Solution Candidates	16
3.2	Technical Overview	16
3.3	Module Design	17
3.3.1	Part-Of-Speech Tagger	17
3.3.2	Retrieval Engine	19
3.3.3	Grammar Realizer	20
3.3.4	Probabilistic Evaluator	22
4	Experimental Evaluation	24
4.1	Text Similarity and Sentence Compatibility	24
4.2	Automated Sentence Evaluation	25
4.3	Evaluation With the Riu System	29
4.4	Evaluation of the Probabilistic Database N-gram Size	32
5	Conclusions	36

Chapter 1

Introduction

Natural Language Generation (NLG) has been a field of research in artificial intelligence since the 1950s[1][2]. Template-based text generation is a successful technique in NLG. These text generation systems use annotated templates with gaps that are filled by the system with the underlying information that needs to be communicated. In this thesis, we have focused on an alternate method: instead of templates, we will use complete sentences, which will be automatically modified by the system to convey the desired information. This method provides great advantages, such as not requiring defining templates, but at the same time poses additional problems.

In this thesis we briefly introduce the general concepts of Natural Language Processing (NLP) and Natural Language Generation (NLG). We continue discussing problems that poses text modification and then proceed to describe a conceptual model and system design to overcome these difficulties.

We deal with the problem of how to generate new textual content by modifying previous sentences. The system described in this thesis focuses on replacing part of a sentence by a given text, while yielding a syntactically correct and semantically coherent sentence.

In this thesis we will suggest using a relatively simple grammatical realizer to improve the generated results of text replacement by using Case-Based Reasoning (CBR) and Search-Based (SB) methods. We also describe automatic solution candidate evaluation using a Probabilistic Evaluator (PE).

The methods described are applied to the domain of computational narrative systems. Specifically, we have worked with Riu, an analogy-based interactive narrative system.

We have explored different methods and we provide experimental results to compare and evaluate the different modules implemented to support the theoretical hypothesis in this thesis. We have used human revised and computer generated datasets to run the experiments described and to support the claims of this thesis.

This document is structured as follows. Section 2 provides an overview and background information in NLP and NLG; and describes the current state of the art of different techniques and methods discussed in this thesis. Section 3 starts with the problem statement followed by the conceptual model, a technical

overview and the design of the system. Section 4 provides some example problems and proceeds with the experimental results obtained with the system. Finally, section 5 states our conclusions, discusses related work and describes lines of future research.

Chapter 2

Background Information

In this section we will provide some background on the several areas of research from where we have drawn in order to complete the work presented in this thesis. Namely, we will report on the areas of Natural Language Processing (NLP) and Natural Language Generation (NLG). We will start by providing some definitions and background concepts used in NLP and NLG that appear in this thesis.

2.1 Definitions and Concepts

In this thesis we make extensive use of linguistic concepts and specific vocabulary. NLP bridges the fields of computer science and linguistics. This first section intends to introduce readers from both technical and non-technical backgrounds into the basic concepts that will be used later in this document.

2.1.1 Language, Natural Language and Computational Linguistics

There are some concepts that one should be familiarized with when working with NLP and NLG.

First of all, **language** can be defined as *a system of objects or symbols, such as sounds or character sequences, that can be combined in various ways following a set of rules, especially to communicate thoughts, feelings, or instructions*¹.

Then, **natural language** is defined as *a language that has evolved naturally as a means of communication among people*². This definition can be further specified for the field of computer science as *a computer language whose rules reflect and describe current rather than prescribed usage; it is often loose and ambiguous in interpretation, meaning different things to different hearers*³.

¹The American Heritage® Science Dictionary Copyright ©2005 by Houghton Mifflin Company. Published by Houghton Mifflin Company.

²Collins English Dictionary - Complete and Unabridged ©HarperCollins Publishers 1991, 1994, 1998, 2000, 2003

³McGraw-Hill Dictionary of Scientific & Technical Terms, 6E, Copyright ©2003 by The McGraw-Hill Companies, Inc.

Linguistics is *the scientific study of language, covering the structure (morphology and syntax), sounds (phonology), and meaning (semantics), as well as the history of the relations of languages to each other and the cultural place of language in human behavior*⁴. And finally, **computational linguistics** is *the use of digital computers in linguistics research*⁴.

2.1.2 Sentences, Phrases and Words

There are several linguistic concepts used in this thesis that may need clarification to avoid confusion.

A **sentence** is *a grammatical unit that is syntactically independent and has a subject that is expressed or, as in imperative sentences, understood and a predicate that contains at least one finite verb*⁵. A **phrase** is a related concept to sentences defined as *two or more words in sequence that form a syntactic unit that is less than a complete sentence*¹.

Phrases and sentences are composed of individual elements commonly called words. A **word** is *one of the units of speech or writing that native speakers of a language usually regard as the smallest isolable meaningful element of the language, although linguists would analyze these further into morphemes*². Words can be classified into different linguistic categories[3] and the same word can be placed in different categories depending on its context. Individual words can be tagged with the class they belong to in a specific context, these tags are called **Part-Of-Speech** (POS) tags[4]. This process is called parsing. To **parse** means *to break (a sentence) down into its component parts of speech with an explanation of the form, function, and syntactical relationship of each part*¹.

Just for the sake of clarification we will give some examples.

"*Love is composed of a single soul inhabiting two bodies.*" (Aristotle) is a sentence whereas *a single soul inhabiting two bodies* is a phrase. In this context the word *love* is a noun. The same word can act as a verb in a sentence like "*I love Alice.*". Using standard POS tags[3], this sentence could be parsed as "PP VBP NP SENT" where PP stands for personal pronoun, VBP for verb in present form, NP for proper noun; and SENT stands for an end-of-sentence punctuation mark.

2.1.3 Language Modeling

Language modeling is the statistical representation of language and is a method widely used in many NLP and NLG applications[5].

N-grams are a common and successful tool for language modeling[6]. An N-gram is a subsequence of n items from a given sequence. N-gram models are a type of probabilistic model used in various areas of statistical NLP[6].

The sentence "*I love Alice.*" is a sequence of words. This sentence could be represented as is, as a sequence of unigrams or 1-grams, where each word would be one item. The same sentence expressed as

⁴Britannica Concise Encyclopedia. Copyright ©1994-2008 Encyclopædia Britannica, Inc.

⁵The American Heritage® Science Dictionary Copyright ©2005 by Houghton Mifflin Company. Published by Houghton Mifflin Company.

bigrams or 2-grams would be "*i love*", "*love Alice*" and "*Alice .*". The N-grams can be in turn encoded into some kind of numerical representation and are often represented as conditional probabilities to predict the most likely word after a certain sequence.

2.1.4 Linguistic Realization

Realization is defined as *the act of realizing or the condition of being realized*¹. In turn, to realize is *to bring into reality; make real*¹. In the context of this thesis, realisation is a subtask of NLG, which involves creating an actual text in a human understandable language. Different kinds of realizers are usually found in different NLG systems, at the very end of the text generation process[7].

2.2 State of the Art

NLP has had successful applications in the areas of information extraction or machine translation[1]. The history of NLP generally starts in the 1950s[2]. Some notably successful NLP systems were developed in the 1960s but those early systems use almost no information about human thought or emotion[8].

Up to the 1980s, most NLP systems were based on complex sets of hand-written linguisticrules. Starting in the late 1980s, however, there was a revolution in NLP with the introduction of machine learning algorithms for language processing[6].

Recent research has increasingly focused on unsupervised and semi-supervised learning algorithms. Such algorithms are able to learn from data that has not been manuallyannotated with the desired answers, or using a combination of annotated and non-annotated data[6].

The most common use of NLG is to create computer systems that present information to people in a representation that is accessible and easy to comprehend. Applications for NLG range from bringing the internal representation used by expert systems to a human understandable form for describing graphical data in a text format[9].

The problem of programming computers to produce natural language explanations and other texts on demand is still an active research area in artificial intelligence[10].

In NLG, simple template-based approaches have been exploited to a great extent with successful results[11]. "Canned text" and template-based solutions are appropriate for some situations, and may even be psychologically realistic up to a point. However, these simple approaches have limitations. The main drawback of canned text systems is the necessity of creating a large base of templates and annotate those accordingly for proper selection and replacement. The resulting systems are relatively inflexible, and all content has to be anticipated ahead of time. They also tend to work best where the output is brief [9].

Modern text generation systems are designed to produce fluent multiparagraph text in response to a goal presented to the system. A common architecture consists of four major modules: a knowledge module which can perform domain-specific searches for knowledge relevant to a given communication goal; a text planning module which can organize the relevant information, decide what portion to present

and decide how to lead the reader's attention through the content; a sentence generation module based on a large systemic grammar of English; and an evaluation and plan-perturbation module which revises text plans based on evaluation of text produced[10].

The latest trends in NLP involve using statistical methods and machine learning algorithms along with leveraging the increasing power of current computers to use big datasets[6][12].

In this thesis we researched the current state of the art for Case-Based Reasoning (CBR)[13] and specifically textual CBR[14]. The work found was focused on NLP. We researched the different phases of the CBR workflow for NLG. Methods for information representation were studied for case and problem description[15][16]. Several text similarity[17][18][19] and graph matching for sentence representation methods[20][21][22] were researched for the retrieval phase. We also studied linguistic-based text realization systems[11], knowledge database-backed systems for NLG[23] and statistical systems applied to NLP and NLG[12][6].

We researched the current state of textual corpora and datasets. One of the most widely used datasets for NLP and NLG is the Penn Treebank [3] as an annotated corpus. Other popular corpora are the Linguistic Data Consortium Gigaword or Google Books N-grams[24].

One of the basic steps for NLP is phrase chunking and POS tagging. Several tools are available for these tasks. The Stanford Parser has three parsers: a high-accuracy unlexicalized probabilistic context-free grammar; a lexicalized dependency parser and a factored model. For parsing English language, the Stanford Parser uses the Penn Treebank corpus and provides three views for output: a context-free phrase structure grammar representation; a typed dependency representation and POS tagged text[25]. A simpler, also freely available option is the TreeTagger by the Institute for Computational Linguistics of the University of Stuttgart[26].

Online knowledge is also necessary for most NLP and NLG tasks[23]. Several dictionaries and tools are available. Wordnet from Princeton University[27] is a famous knowledge database freely available. Several other projects have been created both as standalone tools like ConceptNet from the MIT or online community based projects like DBpedia.org.

There are toolboxes that incorporate the previous tools and a suite of other NLP and NLG related utilities. Some of those freely available are the GATE project ⁶; the OpenNLP project ⁷ or the Natural Language Toolkit⁸.

There is active research into many related fields that promote a very nurturing NLP ecosystem. Just to mention one, a project that has influenced some parts of this work has been the Carnegie Mellon University Sphinx speech recognition system⁹.

A holistic example for NLP and NLG could be talking agents. One of those talking agents is Rollo Carpenter's chatterbot "Cleverbot"[28]. A more prominent example could be IBM's Watson, a

⁶<http://gate.ac.uk>

⁷<http://incubator.apache.org/opennlp/>

⁸<http://www.nltk.org/>

⁹<http://cmusphinx.sourceforge.net/wiki/research/>

machine that recently participated in the popular TV Jeopardy! and successfully defeated two of the best human players in a game involving understanding wordplay, general knowledge and complex response orchestration[29].

Chapter 3

Problem Statement

In this thesis we want to discuss possible solutions for enabling the use of loosely annotated templates for natural text generation using text modification. In this section we will look at the specific problem, our goal, the approaches we researched and the design of a system that implements the discussed methods in a real life environment.

Finding tokens in a string of text and performing replacements is a trivial task for computers. For example one could have the following template:

```
Alice loves Bob.
```

A simple replacement instruction could be stated as follows:

```
Replace "Bob" with "pizza".
```

Therefore the initial sentence would become:

```
Alice loves pizza.
```

The problems of working with natural language will be evident even with simple examples. If we had the same sample template but the following replacement instruction:

```
Replace "Alice" with "I".
```

The output of a blind replacement would be:

```
I loves Bob.
```

It can be easily noticed how the rendered sentence is incorrect due to verb inflexion rules present in the English language.

This sort of situations can be solved with an annotated template with a dependency tree or a typed dependency list[30] that reflects the relationships between components in a phrase and phrase relationships within a sentence. There are common dependency representations that can be used and as long as the grammatical realizer satisfies the requirements of the dependency tree the surface form of

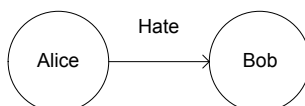


Figure 3.1: Graph-encoded information.

the generated sentence will be correct. For example, if we had the same initial template annotated as follows:

```

(S
 (Subject (NNP Alice))
 (VP (VBZ loves)
 (Subject (NNP Bob)))
 (. .))
nsubj(loves -2, Alice -1)
dobj(loves -2, Bob -3)
  
```

If we wanted to fulfill the same replacement and we provided the following replacement instructions:

```

Replace Subject: "Alice" (proper noun) with Subject "I" (pronoun).
  
```

A dependency checking grammar realizer would proceed by checking the dependencies[30] on the target sentence and applying the linguistic rules implied by the replacement particle therefore would yield a correct output:

```

I loves Bob.
I love Bob.
  
```

This is a really great solution for many systems that work with structured knowledge about entities, properties and relationships; and can map the internal representation with the template sentence tree. For example, a system holds the information encoded in a graph as shown in Figure 3.1. If we assume that the system correctly maps the graph with the sentence we are using as example and that it also maps the corresponding entities, the grammar realizer could bring the information into surface form and maintain grammatical correctness:

```

Alice hate Bob.
Alice hates Bob.
  
```

There are many parsers that can build a dependency tree out of a sentence. The format shown in the previous example is inspired by the output of the Stanford Parser [25].

The problem with natural language, as stated in the introduction, is that there are ambiguous constructs. For example, the following sentence:

```

A kind of dense rock.
  
```

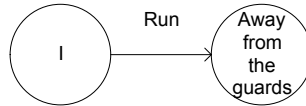


Figure 3.2: Graph-encoded information.

The construct "kind of" in front of an adjective can be interpreted both as a categorization or a colloquial degree modifier. Therefore two very different parse trees can be obtained:

```
(NP
(NP (DT A) (NN kind))
(PP (IN of)
(NP (JJ dense) (NN rock)))
(. .))
```

or

```
(NP (DT A) (JJ kind of) (JJ dense) (NN rock) (. .))
```

These ambiguities are common in English and many other languages. The only safe way around this problem is to use annotated templates.

Finally, even in the case that the template annotation issues are solved and that the grammatical realizer has full linguistic knowledge embedded, it is still due to deal with semantic related problems. A typical example would be Noam Chomsky's example: "Colorless green ideas sleep furiously"[31]. A grammatically correct sentence semantically nonsense.

If we had the in-memory representation shown in Figure 3.2, we could envision a system that has some sort of inaccurate matcher that will request the grammatical realizer to render in surface form the previous information and the following template to be used for rendering.

```
(Subject: I) (Action: walked) (Descriptor: slowly) (Target: away from the
shop).
```

As shown in Figure 3.3, apparently, both the template and the mapping make sense, but the template contains an additional descriptor that the matcher might not have had in consideration, therefore the following output may seem odd:

```
I ran slowly away from the guards.
```

This sort of issues might be solved by working on a matching engine that has access to a comprehensive database of information to check against. In this thesis, we base our work assuming that the required information is not available in the templates and therefore a semantic check is needed.

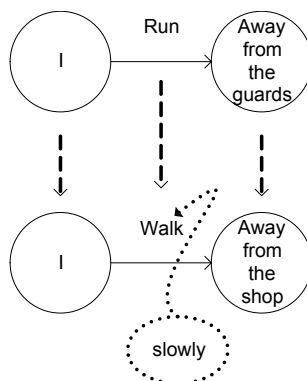


Figure 3.3: Mapping between graph-encoded information and an annotated template.

3.1 Conceptual Model

In this thesis we describe a system composed by 2 methods to solve text modification problems. There is a Case-Based Reasoning (CBR) method and a Search-Based (SB) method. The results of both methods are ranked using a Probabilistic Evaluator (PE) and the final solution is given. A visual overview is shown in figure 3.4. The figure uses standard workflow diagram notation. The information is processed from the inputs of the system at the very top to the final solution at the bottom. The input of the system is represented by a parallelogram labeled as "Problem". Processes are represented by rectangular boxes and preparation processes are represented by hexagonal boxes. The triangles indicate the separation of the workflow into two parallel processes and the final convergence into a solution. We used rounded boxes to represent the different modules containing related processes.

The system described in this thesis takes as its input a problem consisting of a phrase in raw text to be used as a base template and the desired replacements. In order to assist the system identifying the Part-Of-Speech (POS) particles of the replacements, the original raw text context of the replacement token is also feed to the system. The expected output will be a syntactically correct and semantically coherent sentence that renders the input template phrase with the indicated replacements and conveys the information as instructed by the mapping engine.

An example problem is shown in Figure 3.5. In this figure rectangles represent sentences and the linked ovals represent the phrase tokens that will be replaced. The example could be defined as follows, starting with the following template:

"Alice wants to have a pet dog."

And request the following replacements:

"a pet dog", "a pet bird", "Alice used to have a pet bird."

"Alice", "The daughters of a sad pianoman", "The daughters of a sad pianoman sing many happy songs."

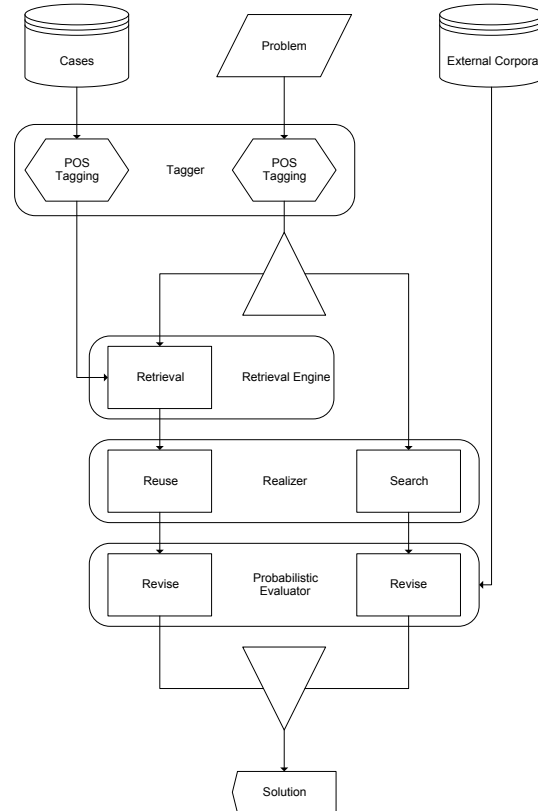


Figure 3.4: Overview of the system.

Once replaced the output is rendered as follows:

"The daughters of a sad pianoman wants to have a pet bird."

The expected output instead would be:

"The daughters of a sad pianoman want to have a pet bird."

The system starts by running the raw text sentences through a POS tagger. Once tagged, the problem description will be developed into top level task requirements. These requirements are extracted from the POS-tagged replacements and specify the strings of the POS particles to be replaced and to be used as replacements. The system starts by performing the replacements and then checking for additional constraints to be satisfied as for the top level task requirements defined by the problem description. These additional constraints are determined using the CBR method explored in this thesis and described below.

The CBR method will load from a database of cases, a shortlist of example cases containing similar top level task requirements and similar base sentences. The example cases in the database are encoded along with Text Transformation Routines (TTR); the routines required for satisfying the constraints of the top level task.

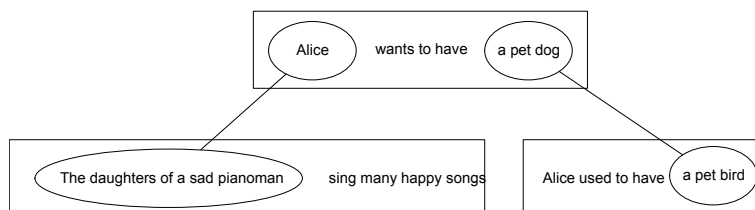


Figure 3.5: Problem example.

TTR are simple textual routines that the grammar realizer can perform. TTR work at a low language level and provide functionality like inflexion or word removal. For example the DET TTR will add a determiner. Some TTR are in turn hierarchical and polymorphic, for example the PLURALIZE TTR, when executed over a phrase will break down into executing itself over each POS particle. When PLURALIZE is executed over a noun will pluralize it but if it is executed over a verb in 3rd present indicative form, will turn it into regular present indicative form.

Each TTR execution yields a single modification to the base sentence. For every generated sentence a copy is stored as a solution candidate. The TTRs are executed sequentially and each modification is cumulative, rendering a slightly modified phrase for each TTR.

Alternatively, the problem is run through a SB method. This takes advantage of the same grammatical realizer. The SB method will expand a depth-bound tree of all possible language level modifications for each of the replaced sentence's particles. It runs every available TTR for each POS particle. Each leaf is considered as a solution candidate.

Eventually, the system has a list of solution candidates. The solution candidates are then ranked using the Probabilistic Evaluator (PE). The system should detect and discard both syntactic and semantic inaccuracies caused by non-applicable TTRs carried previously.

3.1.1 Case-Based Reasoning for Natural Language Generation

CBR for textual NLG poses major problems in the 4 stages of the CBR workflow[13] starting with the case and problem description itself.

Case and Problem Description

In natural language, it is very challenging to find any encoding more compact than the raw representation of the original text that will keep all the semantic and syntactic information. On the other hand, any additional metadata may be used in order to prune and reduce the search space on retrieval. Therefore handling the case database poses interesting challenges for structure and performance.

We decided to encode problems into objects containing the source sentence and a list of the replacements with the context of each token to be replaced. On the preprocess phase, we use a POS tagger to tag both the sentence and the replacement tokens, discarding the context sentences.

The cases are stored as triplets of a source sentence, a modified target sentence and a list of desired replacements with the associated TTR required to satisfy the constraints for each replacement. In order to retain the maximum amount of information in the case data base, each sentence is stored in raw text; the replacements list only stores the POS particles for the replaced and replacement tokens; and the TTR and context of the applied transformation, also encoded in POS form. To ensure a successful matching on retrieval, the source sentences need to be authored not to contain ambiguous POS particles.

For example, we could have a case in the database where the original sentence "*The girl has always wanted to be a painter.*" has had the token *girl*: a singular common noun (NN using the TreeBank[3] notation); replaced by *girls*: a plural common noun (NNS also using the TreeBank[3] notation). In this example case, the replacement noun is coordinated with the auxiliary verb *has* and the second common noun *painter* and therefore this creates a number constraint. This constraint can be satisfied by applying the PLURALIZE TTR. As we described previously, the PLURALIZE TTR is hierarchical and can be applied at sentence or phrase level, therefore the case in the database could be encoded as follows:

```
The girl has always wanted to be a painter .
The girls have always wanted to be painters .
NN>NNS: pluralize /S/*
```

The problems will be constructed from raw text and parsed into a compatible form for the retrieval process. The POS-tagged contexts of the replacement tokens will be used to generate top level task requirements that match the constraints defined previously. In the retrieval phase, constraint satisfying TTRs will be loaded into the problem.

Retrieve

The main challenges found in the retrieval phase are both the case and problem representation and the similarity function used for retrieval. Also worth of mention is the fact that on retrieval one must choose between checking for syntactic or semantic matching as checking for both may be overly complicated.

For the retrieval phase we decided to use a multi-step search process to increase both performance and accuracy. The first step ranks the similarity of the cases in the data base against the current problem using the POS information in the replacements list. The N top ranked cases are then evaluated for similarity using the full tagged sentence against the full tagged problem. The similarity function evaluates both syntactic structure and semantic content for certain matched POS particles.

Reuse

A great difficulty in the reuse phase stems from the fact that the mappings between the problem and retrieved cases tend to be fuzzy and ambiguous. On top of that, unlike other domains where functions can be clearly defined, in the natural language domain, functions to be applied for adaptation tend to be grammar dependent and conditioned to context rules and exceptions.

A hierarchical plan is created for each problem. Each TTR from the retrieved candidates is appended at the top level of the plan and the context for the plan is the full problem's sentence. Each subtask in the plan consists of a TTR and a restriction on the scope for the TTR to be applied to. The TTR is then applied to each of the elements of the sentence that matches the defined scope. The grammar realizer is queried and each POS may provide a specific implementation for the function or task being executed. Each function or task yields a possible solution candidate which is stored for review.

Revise

In some domains, evaluation functions can be defined to yield a measurable success of the proposed solutions. Although there has been a lot of research into natural language processing, automated tools beyond simple grammar rule checking and basic semantic extraction cannot be relied on. The nature of natural language poses cases that may be ambiguous even to a human reviewer.

The revise process is performed by the automated PE evaluation module. For each candidate solution, we use a sliding window of N words to check for frequency and probability against an N-gram database. This strategy tries to identify both syntactical errors and semantic incoherence. The possible solutions are then sorted and the top ranking candidate is returned as the proposed solution.

Retain

The challenges of the retain phase are contained among the challenges already described.

An automatic retain phase has not been included in this project due to the challenges of selecting distinctive candidates. However, a human-reviewed selection can be made and the results manually included in the case database.

3.1.2 Search-Based methods for Natural Language Generation

The SB method defined in this thesis is conceptually simple. We had to overcome several challenges when designing the search concept to be applied to the NLG domain.

From a starting sentence, which is used as the base state, a set of functions are used to generate subsequent states. Subsequent states are variations of the starting sentence resulting from applying small text modifications. The SB method involves evaluating every possible candidate and expanding its subsequent states. Although a sentence can be seen as a starting state, search for natural language is an ambiguous concept because the functions to generate the subsequent candidates are not clearly defined. The design of the SB method defines a subset of simple linguistic rules and uses these as functions to get to new states. The functions are actually the TTRs made available by the grammatical realizer and are applied to every POS particle to develop the search tree.

Using blind search in the domain of NLG is a great difficulty as the possible tree for the simplest sentence may become extremely large very easily.

In order to minimize the search space, we only expand the root branches from a sentence and follow only the first branch, expanding each subsequent branch once, and exploring the particles in the order they are replaced by the previous task.

Each sentence generated is stored as a solution candidate and the candidates are ranked in the same manner they are in the revise phase of the CBR described earlier.

3.1.3 Probabilistic Evaluation of Solution Candidates

The final step for both methods explored in this thesis is a revision phase that involves an automatic reviewer to rank the solution candidates and determine a final solution.

The evaluation approach we wanted to design had to be language-agnostic so we wouldn't have to deal with encoding complex language-dependent rules.

We use N-grams as Markov chains in order to evaluate the plausibility of a generated chain of words. This method relies heavily on an external N-gram database. N-gram databases are widely available or can be built easily with a large corpus.

3.2 Technical Overview

The project described in this thesis has been implemented as a module to improve the output of Riu[32].

Riu is a text-based interactive narrative system that uses analogy to generate stories in order to explore new narrative spaces. It creates stories about a robot character Ales, who has initially lost his memories. Similar to the protagonist of Pan's Labyrinth, Ales constantly oscillates between his recovering memory world and the main story world (real world). The two worlds not only share parts of the structure, but also influence each other. Events happening in the memory world may impact the development in the real world. Riu explores the same story world as Memory Reverie Machine (MRM). Riu focuses on computational analogy with a force-dynamics based story representation. Riu is composed of four main modules: a story engine, a memory retrieval component, the Ales module, and a computational analogy component, the Structure Mapping Engine (SME).

In addition to the previous four modules, Riu contains two repositories of authored data: a main story (consisting of a graph where each node is a scene, and each link is a possible action that Ales may take) and a collection of past memories.

The computational analogy module, SME, is used both by the memory retrieval and by the Ales module to find mappings between scenes.

The SME finds analogies between different scenarios and when a match is found, Riu replaces the tokens in the existing text with the mapped analogies. The system has some semantic information about actors and objects but the replacements are performed blindly, therefore some syntactic inconsistencies

Table 3.1: Example problem from Riu.

Target sentence	Alice wants to have a pet dog.
Token to be replaced	Alice
Replacement	The daughters of a sad pianoman
Context of the replacement	The daughters of a sad pianoman sing many happy songs.
Token replacement	The daughters of a sad pianoman wants to have a pet dog.
Expected output	The daughters of a sad pianoman want to have a pet dog.

or sometimes semantic incoherence may appear. Once the entities are mapped the output is rendered by token replacements. A great difficulty is the fact that in Riu, a mapping token can represent a single word, a phrase or a full sentence.

Riu performs one replacement at a time. One of the replacements of the example problem discussed previously is described in Table 3.1.

As it can be observed in the example, blind token replacement may result in non-correct or non coherent sentences.

The system described in this thesis takes as input the source sentence, the token to be replaced and the matched token to be used as replacement. The system will take advantage of the context sentence of the token to be replaced being available to syntactically tag the tokens with the correspondent POS particles.

The goal of the project is to attempt to create the most compelling results to be displayed to the user within the interactive storytelling environment. The interaction of the system described in this thesis with the Riu narrative system is shown in figure 3.6. In this figure, the internal Riu modules are represented by rounded boxes. The rectangular boxes separate the Riu core from the system described in this thesis and the different data sources and user interface.

3.3 Module Design

3.3.1 Part-Of-Speech Tagger

The POS tagger is a wrapper for the external TreeTagger[26], a language independent part-of-speech tagger by the Institute for Computational Linguistics of the University of Stuttgart. It performs probabilistic part-of-speech tagging using decision trees. We also take advantage of the stemming capabilities of TreeTagger[26].

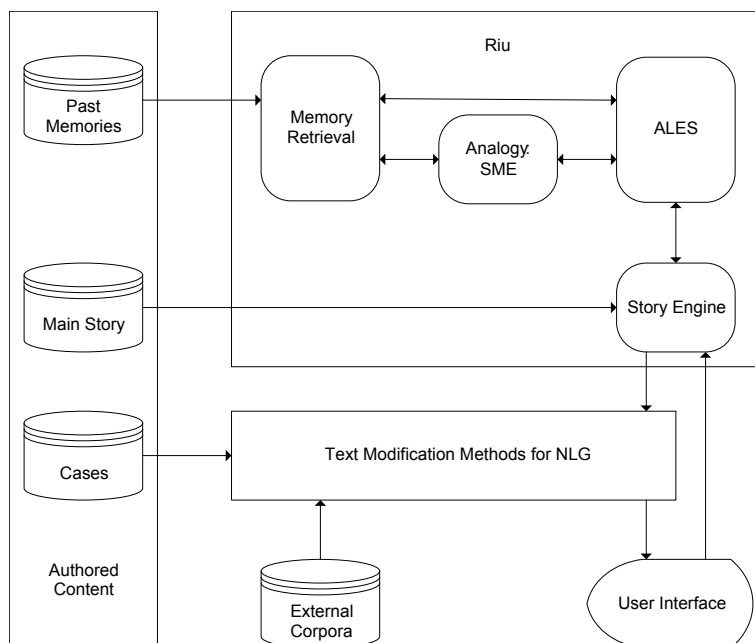


Figure 3.6: Integration with the Riu narrative system.

We use the English parameter file, trained on the Penn Treebank annotated corpus, released through the Linguistic Data Consortium[3].

For efficiency sake, the module uses an internal cache of tagged sentences but can be easily configured to work online with the external TreeTagger executable. The output of the TreeTagger executable may not be accurate in some cases which may pose ambiguous even to a human reviewer. The cache files have not been authored in any way in order to be loyal of one of the goals of the project of minimal human interaction.

Below is a sample run of the POS tagger over the sample sentence "*Alice used to have a pet dog.*".

Alice	NP	Alice
used	VBD	use
to	TO	to
have	VB	have
a	DT	a
pet	JJ	pet
dog	NN	dog
.	SENT	.

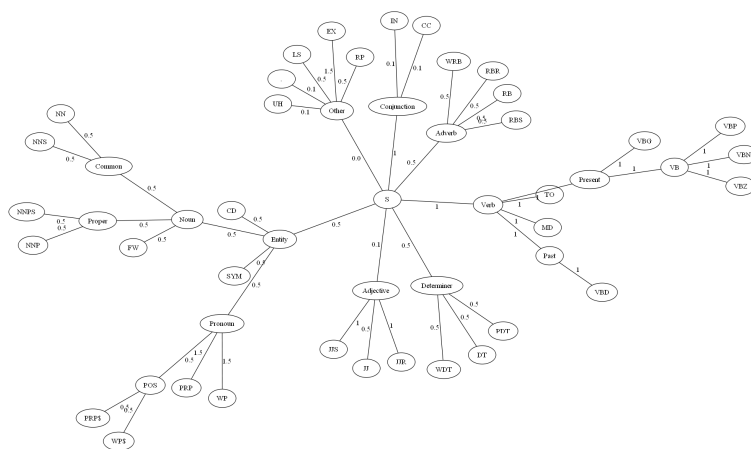


Figure 3.7: Graph-encoded costs for POS particles.

3.3.2 Retrieval Engine

The retrieval process is one of the most complex processes of the system. The retrieval process consists of two phases. The first phase prunes the number of cases to be checked in the case database and the second phase ranks and sorts the shortlist of candidates.

In the first phase, only a POS-tagged string of the task requirements is used. To build the POS-tagged string of the task requirements, the raw text of the problem's phrase and the raw text of each case in the database are POS-tagged. The substrings of the replacement tokens are used for phrase chunking and to extract a POS string. The POS string is used as a simple representation of the task requirements and the string form enables us to use our custom String Edit Distance (SED)[19] function.

For the second phase, we use the full version of our customized SED algorithm to compute the similarity between the current problem and each of the selected case candidates. Our SED algorithm uses POS-tagged words as tokens for the string components along with the raw word. An undirected weighted graph was built with the different linguistic components of a phrase and the POS particles used by TreeTagger were added as leaves to the graph. The complete graph is shown in Figure 3.7. The cost functions use a cost table built by traversing the paths from each leaf in the graph.

We estimated the cost of inserting and deleting a component by adding a root element to the graph. The insertion and deletion cost is calculated from the leaves to this root element. The arcs are weighted with relative numeric costs estimated by studying the impact of adding and removing each possible linguistic component and POS particle from a base sentence. For example, we assume that inserting or removing an adjective or an adverb from a sentence has a smaller effect than adding or removing the subject or the main verb, therefore the arcs have higher values for the latter.

In the first pruning phase, only the previously described method is used. In the second phase, the selected candidate cases are run against the current problem through the full SED algorithm to compute the similarity coefficient that will be used to rank and sort the cases. The modification cost is computed

in the same manner, but two additional cost components are calculated and the total modification cost is obtained by a weighted summation. This process enables us to fine tune the selection of the most compatible case when there are syntactically identical candidates by using two different semantic metrics. Stemming is used to lookup the semantic content of the string components for primordial compatibility using the Regressive Imagery Dictionary (RID) and for conceptual coordination using Wordnet.

Below is a sample run of the similarity ranking of a set of sentences against the sample sentence *He was even learning to be a painter.*

```
He was even learning to be a painter. 0.0043
He was even learning to be a dancer. 0.0143
He was even starting to be a painter. 0.0147
He was even learning to paint. 3.3033
He was even walking against the crowd. 5.2221
```

Regressive Imagery Dictionary

The RID is a coding scheme for text analysis that is designed to measure "primordial" and conceptual or secondary content. The RID contains about 3,000 words grouped into categories that are themselves classified as primary, secondary, and emotional. For each category words are grouped under similar concepts in a hierarchical tree¹².

The RID has been observed to yield a valid index of primordial or dedifferentiated thought in a variety of contexts[33].

Although the RID has been used for sentiment and intention analysis it is used in this thesis to check distance between words in the hierarchical word tree.

Wordnet

Wordnet is a lexical database for the English language developed by the Cognitive Science Laboratory at Princeton University. Wordnet groups English words into sets of related words called synsets and provides short descriptions and relationships between synsets.

Wordnet has been widely used for many NLP and even NLG. Although we initially used synonyms and antonyms with positive and negative feedback, we determined that using the coordinated terms of both the source and target words was a broader, more effective approach when checking for compatibility.

3.3.3 Grammar Realizer

The grammar realizer module has been designed as a simple yet extensible module that stands at the core of both CBR and SB approaches of this thesis.

¹<http://www.kovcomp.co.uk/wordstat/RID.html>

²<http://www.kovcomp.co.uk/wordstat/RID.html>

The grammar realizer module has been designed to implement a high-level language-agnostic interface. One of the design goals of the module was to encode minimal linguistic information and maximize the use of statistical methods instead of hardcoded language-dependent rule information.

The module understands a subset of the POS tags of a given sentence and for each may provide alternative TTR that enable transformations over the whole sentence, a limited phrase or a single POS particle. We have coded simple methods for the English language that enable number based transformations; semantic replacements; noun, pronoun and determiner transformations; and verb inflection.

Polymorphic POS objects enable each transformation to be made available and modified. We have coded subclasses for nouns, verbs, adjectives, determiners and punctuation symbols. A superclass takes care of phrase-level functions such as capitalization.

Any of the exposed TTR can be requested to be applied over a full sentence, a phrase or any single POS particle. A dedicated language has been designed to determine the context for an applied method. The language is human readable and is encoded along with the cases in the CBR database as described previously. Below there is a sample run of the grammatical realizer over the sample sentence *Alice wants to have a pet dog.*

```
Alice wants to have a pet dog.  
pluralize/S/**  
Alices want to have pet dogs.  
singularize/S/**  
Alice want to have pet dog.  
remove/S/JJ  
Alice want to have dog.  
det/S/NN  
Alice want to have a dog.
```

In the previous example we can observe both the capabilities of hierarchical TTRs and also some of the problems encountered. We consider it relevant to mention that some of the problems observed in the experimental evaluation may be due to an incomplete grammar realizer instead of design faults in the CBR or SB approaches. If we enhanced the grammar realizer, the overall output would be substantially better.

The exposed functions and methods use several external knowledge tools to provide its services. These are the MontyLingua toolbox, the Wordnet lexical database and the Inflect Python package.

MontyLingua

A heavily patched version of the Python MontyLingua linguistic toolbox by Hugo Liu is used for verb inflexion capabilities.

Wordnet

The Wordnet lexical database by Princeton University is used for semantic capabilities like synonym and antonym replacements.

Inflect

The Inflect Python package by Paul Dyson is used for additional minor capabilities like number based transformations.

3.3.4 Probabilistic Evaluator

The PE module is an important piece in both approaches described in this paper. It plays a specifically relevant role in the SB approach.

The evaluator has been designed as a self contained, pluggable module so multiple evaluators can be switched or combined.

The current evaluators use each a full featured N-gram database. We determined that using a sufficiently large database we could account for both correct syntax and semantic coherence. Each individual evaluator in the PE module depends on its N-gram database and therefore is language dependent.

The evaluator takes for input a phrase as a raw string. The string is preprocessed and parsed into a sentence. The sentence is then tokenized according to the format of the N-gram database. Certain tokens are discretized into categories, like numerals and ordinals. Then, the module uses a sliding window of n words to check for frequency and probability against an n -gram database. The size of the window is determined by the maximum length of the sequences in the database.

In a context-free environment, the evaluators have been observed to perform better the longer the N-grams in the database. On the other hand, increasing the length of the N-grams in the database increases the file size exponentially and the processing power required for queries..

It was observed that giving a numeric score to a single sentence was not a trivial task and lead to ambiguous scores in many cases. In order to sort solution candidates, the evaluator module uses all the available N-gram occurring frequency or probability values and compares each sentence using the full list. The lists are sorted in ascending order and each element is compared against the same ordinal in the rest of the lists from the current set of sentences.

We studied two evaluators using different databases in order to be able to compare and benchmark each other. We used a frequencies database (Google N-grams) and a probabilities database (Gigaword Arpa).

Google N-gram

This evaluator uses a language model based on the Google Books N-gram English dataset. The original dataset was extracted from a corpus scanned books. The dataset in use is a custom aggregated

version of the original 4-gram dataset, limited to books between 1980 and 2008, discarding year information, words with non-English characters (with the exception of the diacritic accent) and discretizing numeric values and punctuation symbols.

N-grams are listed one per line and for each, its absolute count. The occurring probability of a single N-gram, is computed in standard deviations from the mean of its parent $N-1$ -gram. For simplicity sake we assume a normal distribution.

The occurring probabilities of each N-gram present in the phrase being evaluated are stored as a list. The list is finally sorted in ascending order.

A sample run of the evaluator over a set of sentences is shown below.

I have a pet dog .	0.03 , 1.23 , 4.52 , 21.0
I have a pet wolf .	-100 , 0 , 0.03 , 21.0
I have a pet hamburger .	-100 , -100 , 0.03 , 21.0

Gigaword ARPA

This evaluator uses a language model based on the Linguistic Data Consortium Gigaword text corpus consisting of 1.200 million words from various sources.

The original corpus was compiled by David Graff and Christopher Cieri. The evaluator uses language model files in the ARPA format created by Keith Vertanen. The files contain monograms, 2-grams and 3-grams and the vocabulary has been limited to the 64.000 top occurring words.

The ARPA backoff format was developed by Doug Paul at MIT Lincoln Labs for research sponsored by the U.S. Department of Defense Advanced Research Project Agency (ARPA)³.

The ARPA file format for N-gram backoff models starts with a header, introduced by the keyword `\data \`, listing the number of N-grams of each length. Following that, N-grams are listed one per line, grouped into sections by length, each section starting with the keyword `\N-gram`, where N is the length of the N-grams to follow. Each N-gram line starts with the logarithm (base 10) of conditional probability p of that N-gram, followed by the words $w_1 \dots w_N$ making up the N-gram. These are optionally followed by the logarithm (base 10) of the backoff weight for the N-gram. The keyword `\end \` concludes the model representation.

The conditional probabilities of each N-gram present in the sentence being evaluated are stored in the list used to rank and sort sentences. The sentence score is calculated as the average of the list.

A sample run of the evaluator over a set of sentences is shown below.

I have a pet dog .	-2.7593
I have a pet wolf .	-3.5635
I have a pet hamburger .	-4.17445833333

³<http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html>

Chapter 4

Experimental Evaluation

In order to evaluate the proposed approaches, we designed 3 different experiments.

The first experiment aims at evaluating the retrieval engine design and the sentence compatibility algorithm. A second experiment explores the performance of the Probabilistic Evaluator (PE). The third experiment tests the design of the system as a whole using output from the Riu narrative system.

Some experiments contain ambiguous or numerically difficult to analyze linguistic and semantic information. The results analyzed were benchmarked against human authored responses. The responses were collected from a survey on a group of selected individuals from different backgrounds, all with college-level education and working knowledge of the English language.

One more example experiment is described at the end of this chapter as an addendum to ratify on the importance of the N-gram database and validate the design choice of a 4-gram ceiling.

4.1 Text Similarity and Sentence Compatibility

We wanted to test the performance of our sentence similarity algorithms for the retrieval engine. The similarity metric takes into account syntactic and semantic aspects and is used by the retrieval engine to retrieve the cases .

The similarity algorithm returns an unbounded numeric result that indicates how different sentences are, being close to 0 for identical sentences. Note that it might not be 0 in all the cases because punctuation is handled separately. Since using an unbounded numeric value to check for compatibility between two sentences gives little information about how similar or dissimilar they are, we decided to use sets of sentences. The sets consist of a base sentence, and then a collection of sentences generated by modifying the base sentence in different ways. The generated sentences are then placed in an ordered list from most similar to less similar sentence compared to the base sentence.

Each set is authored to check for certain features. Sets were crafted to test for basic noun and pronoun compatibility, adjective insertions and deletions, several linguistic modifications of varying degree and

semantic modifications. Nonsense sentences were added. The sorted list order was established by trying to sort the sentences from more similar to least similar to the base sentence; we used the surveyed results from several human subjects to assist us in some of the ambiguous cases. The first set of experiments consists of using a similarity metric to compute a similarity value between each of the sentences in the set to the base sentence, and then sorting the sentences using this value. The closer the obtained order to the order generated by human subjects, the better the similarity metric captures how humans see two sentences as similar. Table 4.1 shows some of the results.

The targets for each set are: *Bob loves Alice.*; *Alice used to have a pet dog.*; *Alice loves pizza.*; respectively.

The experimental results show how the similarity metric correctly groups sentences' proper nouns and primordial related verbs (love, like and hate) but fails to recognize the pronouns substituting nouns and some other verbs (wants and desires; makes and bakes).

In other sets, the similarity metric correctly groups sentences with changes in the main verb and therefore sentence compatibility but fails to correctly weight the impact of adjective changes versus noun changes. Noun compatibility is also observed not to always take into consideration conceptual similarity and some related concepts are ambiguously tied. Some fine tuning on weights and a greater knowledge base is expected to solve the problems.

In overall, the similarity metric groups and sorts the tested sentence sets in the same manner a human would sort them between 60% and 80% of the time.

4.2 Automated Sentence Evaluation

In order to test the PE performance for automatic evaluation, as in the previous experiment, we crafted several sets of sentences. The sets are designed to test evaluators against different grammatical and syntactic phrase features.

There are separate sets to check for grammatical errors and semantic incoherence or other specific tasks. Other sets check for general problems, or focus on dealing with long sentences where phrasal elements are spaced by more words than the size of the N-grams in the database.

As in the previous experiment, each set contains several sentences with some text modification. Each sentence is run through the evaluator and a list of probabilities or frequencies is returned and used to sort the set. As in the previous experiment the original set is sorted with the expected output order.

The internal design of the evaluation process is evaluated by displaying the raw list of probabilities or frequencies along with the results.

A phrase is first tokenized and, if not already so, and then it is converted into a complete sentence by adding the corresponding final punctuation. Then we use a sliding window matching the size of the length of the N-grams in the current evaluator's database. Each N-gram is checked against the database and its corresponding probability or frequency is reported. For the frequency metric, instead of using the N-gram frequency, we use the standard deviation from the (N-1)-gram mean frequency.

Table 4.1: Retrieval and similarity examples.

Not all results are shown.

Expected	Result	Dist.
I love Alice.	Alice loves bob.	0.0023
Alice loves me.	Bob hates alic.	0.0078
Bob hates Alice.	Alice alic alic.	3.0021
I hate Alice.	Loves loves loves.	4.4
Alice Alice Alice.	I hate alic.	4.4
Loves loves loves.	I love alic.	4.4
Alice used to have a little dog.	Alice used to have a little dog.	0.0041
Alice used to have a cute little dog.	Alice used to have a cute little dog.	1.6041
Alice used to have a dog.	Alice used to have a dog.	2.0034
Alice used to have a pet bird.	Alice used to have a pet bird.	0.004
Alice used to have a pet wolf.	Alice used to have a pet wolf.	0.0041
Alice used to have a pet hamburger.	Alice used to have a pet hamburger.	0.0041
Alice used to have a hamburger.	Alice used to have a hamburger.	2.0039
Alice used to have a hamburger daily.	Alice used to have a hamburger daily.	4.0039
Alice wants to have a pet dog.	Alice wants to have a pet dog.	5.014
Alice wants to have a cute little dog.	Alice wants to have a cute little dog.	6.6045
Alice likes pizza.	Alice likes pizza.	0.0052
Alice wants pizza.	Alice hates pasta.	0.0078
Alice desires pizza.	Alice dislikes pizza.	0.0078
Alice makes pizza.	Alice detests pizza.	0.0078
Alice cooks pizza.	Alice hates pizza.	0.0078
Alice bakes pizza.	Alice makes pizza.	0.0102
Alice dislikes pizza.	Alice bakes pizza.	0.0127
Alice detests pizza.	Alice wants pizza.	0.0127
Alice hates pizza.	Alice cooks pizza.	4.4
Alice hates pasta.	Alice desires pizza.	4.4

For example, the following sentence:

```
I have a pet wolf.
```

The output using the Google 4-grams would be:

```
. i have a          21.0154491913
i have a pet       0.0396576915049
have a pet wolf   -100
a pet wolf .      0
```

Note that the evaluator can work in phrase and sentence modes. When the sentence mode is triggered, as in the example, leading and trailing punctuation marks are added when not present. Punctuation marks enable us to check for the likelihood of a standalone sentence. When the evaluator runs in phrase mode, no punctuation marks are added to account for the possibility of the phrase being part of a bigger sentence and no beginning and ending of sentence frames are checked.

After collecting the frequency information, it is sorted in ascending order, therefore the previous sentence would yield the following list:

```
-100,0,0.03,21.0
```

The same procedure is applied to the rest of the sentences in the set and then the set is sorted according to the list. The ordering method will check the lower values, indicating the least probable or frequent N-grams and when identical, will proceed to the next one until the list is exhausted or there is a divergence.

The sets have been run through both evaluator databases. While the first run uses the Google Books 4-gram database of frequencies, the second run uses the Gigaword CMU Sphinx ARPA 1, 2 and 3-gram database of conditional probabilities. Some examples are shown in Table 4.2.

From the experimental results, it can be observed that the conditional probabilities database performs better than the frequencies database (2nd in the league); but also that using a large database of frequencies the output is similar if not better than using a smaller database of conditional probabilities (I run fast).

The results for both experiments range at are around 75% of the expected results from the human annotated sample. The results though vary greatly from one database to the other. With either database, the PE had troubles evaluating sentences that contained fantasy character names and other elements from the stories from Riu. We conclude that using domain-specific corpora could help the results of the PE.

There has been observed curious phenomena that cannot be explained by the evaluator design. Sentences where related words are further apart than the total length of the N-grams, should not be evaluated correctly. This shouldn't impact much when evaluating sentences where short replacements have been made, but it leads us to think that the corpus text of the database can greatly affect the PE.

Table 4.2: PE evaluation examples.

Not all results are shown.

Expected	R. Google	R. ARPA
He plays the piano. He play the piano. He plais the piano.	He plays the piano. He play the piano. He plais the piano.	He plays the piano. He play the piano. He plais the piano.
They play the piano. They plays the piano. They plai the piano.	They play the piano. They plays the piano. They plai the piano.	They play the piano. They plays the piano. They plai the piano.
I run fast. I run slower. I run slowly.	I run fast. I run slower. I run slowly.	I run fast. I run slowly. I run slower.
I have a pet dog. I have a pet wolf. I have a pet hamburger.	I have a pet dog. I have a pet wolf. I have a pet hamburger.	I have a pet dog. I have a pet wolf. I have a pet hamburger.
Our team was 2nd in the league Our team was 2 in the league Our team was hamburguer in the league	Our team was 2 in the league Our team was 2nd in the league Our team was hamburguer in the league	Our team was 2nd in the league Our team was 2 in the league Our team was hamburguer in the league

4.3 Evaluation With the Riu System

A sample problem from a previous paper on Riu [32] has been used to benchmark the results of the system described in this thesis.

The following scene is matched with an unknown scene by SME.

```
Ales had always wanted to be a painter .
He was even learning to be a painter .
But the long hours of his day job leave him very little time to practice
eventually giving up .
```

The scene is processed as 3 different problems defined as follow:

```
Ales had always wanted to be a painter . "be a painter" / "walk against the
crowd"
He was even learning to be a painter . "learning to be a painter" / "walk
against the crowd"
But the long hours of his day job leave him very little time to practice
eventually giving up . "practice" / "walk against the crowd"
```

An artificial context sentence has been crafted to be able to parse the *"walk against the crowd"* phrase: *"Alice has always wanted to walk against the crowd."*

Without our system, the scene would be rendered as follows:

```
Ales has always wanted to walk against the crowd .
He was even walk against the crowd .
But the long hours of his day job leave him very little time to walk
against the crowd eventually giving up .
```

The second sentence result is not grammatically correct. The same sentence, ran through the system would yield the following output.

TARGET

```
He was even walk against the crowd .
```

SOLUTIONS

```
He was even walking against the crowd . (-48.18, 1)
He was even walks against the crowd . (-64.83, 1)
He was even walk against the crowd . (-64.83 Source , 194)
He is even walk against the crowd . (-64.92, 4)
```

This example shows the output for one problem. We ran the same experiment over 700 problems created from different stories and scenarios of the Riu system. The problems created by the Riu system can be very hard as strange mappings occur in SME and therefore some replacements might make little

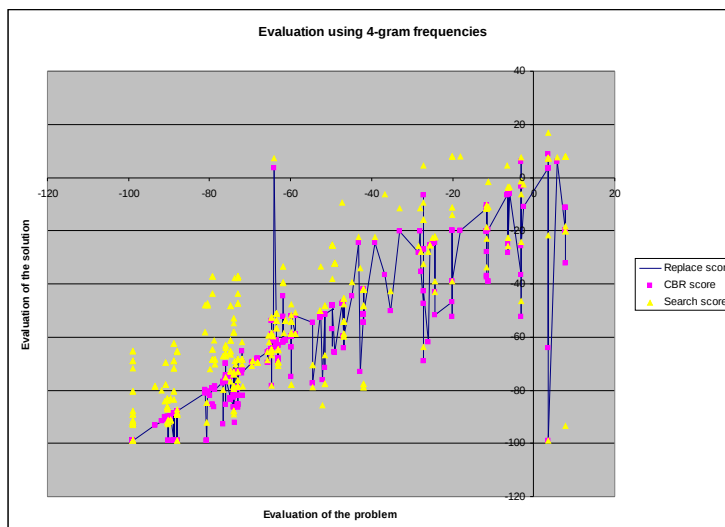


Figure 4.1: Experimental results from the Riu system using the conditional probabilities database.

sense. Some were unsolvable even for the human reviewers that assisted us. 10 additional problems were handcrafted and added into the problem set for benchmarking reasons.

For each problem, we recorded the problem and the base problem’s sentence evaluation using both of the PE databases. Then the replacements were performed and the replaced sentence evaluated and recorded in the same manner. The problems were then solved through both the Case-Based Reasoning (CBR) and Search-Based methods. The process was ran twice, using both of the PE databases. The best ranked solutions were recorded along with the evaluation.

Figures 4.1 and 4.2 show the results of running the full experiment. The numeric score of the evaluated original sentence, as computed with the PE using each database is plot on the x-axis. The scores of the resulting sentences are evaluated using the same method and plot on the y-axis. The replacement score is calculated from the sentence performing the required replacements with no involvement of the system. This is not a continuous value and is plotted as a line for visual differentiation.

The scale used in the figures is the same as described in the previous chapter when describing the databases used by the PE.

In order to numerically evaluate the impact of our methods over the data we discounted the original sentence’s evaluated score to the resulting sentence’s evaluated score. The measurements are summarized in Tables 4.3 and 4.4

It can be observed that when using the frequency database, there is a greater dispersion of the results but when looking at the numeric data, in both cases, the SB method outperforms the CBR method in about 10%.

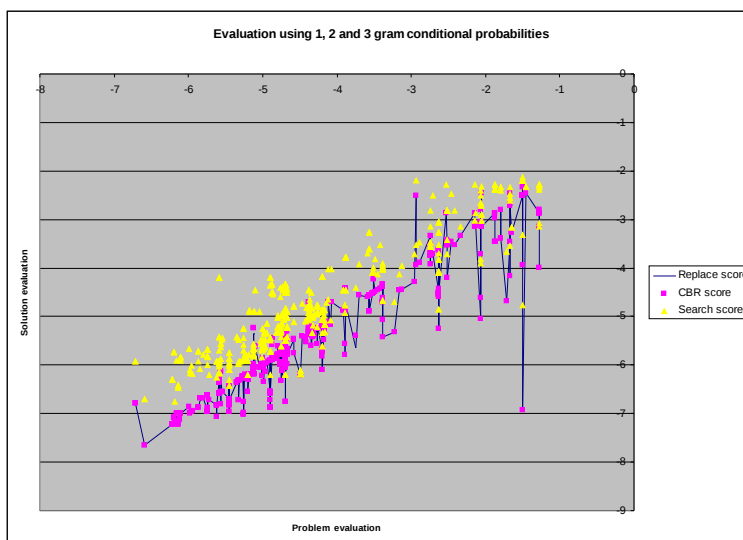


Figure 4.2: Experimental results from the Riu system using the frequencies database.

Table 4.3: Google N-grams (frequencies) summary.

	Replace	CBR	Search
Average	-69.014	-70	-59.3913
Discounted average	-3.59601282	-0.55571795	10.1784487
Discounted standard deviation	5.88128307	1.11695203	10.0000681

The CBR method was expected to perform better, therefore individual tests were carried. We wanted to check the impact of the size of the case database in the results. We ran the first 50 problems through the system using different case databases. The smaller case database is actually a relevant subset of the original case database. The results are shown in Figure 4.3.

Using the automated evaluation system described before we observed a counterintuitive slight degradation of the performance with the larger case database, but results were inconclusive. We determined that the culprit was likely the fact that previously we had to loosen the restrictions imposed by the retrieval engine so it could always load at least 3 cases. This causes the retrieval engine to load some of the cases very frequently and those cases involve the same transformations that then the PE will incorrectly rank higher.

We also wanted to analyze the individual problem output. We selected a few examples that are shown in Tables 4.5 and 4.6.

The main problem found when examining the individual solutions was that sometimes the system, when selecting the solution candidate, would favor some sentence variations, usually created by the SB

Table 4.4: Gigaword ARPA (conditional probabilities) summary.

	Replace	CBR	Search
Average	-5.54576923	-5.52	-4.82
Discounted average	-1.13489744	0.0239359	0.69939744
Discounted standard deviation	0.24822108	0.04328471	0.31596021

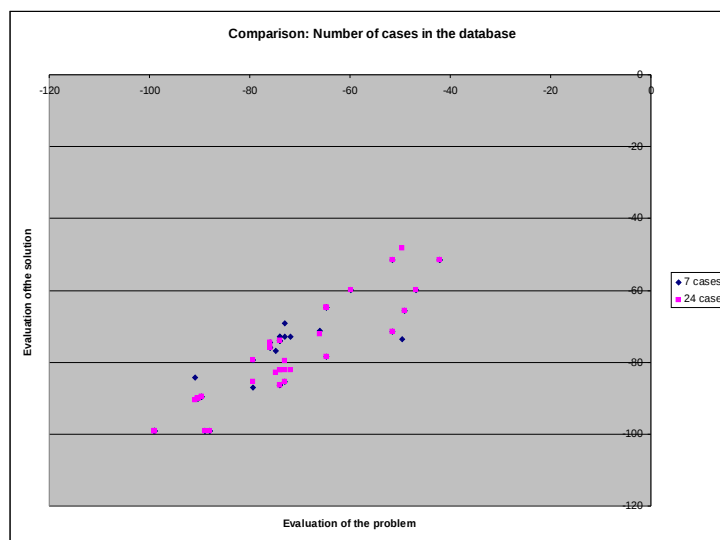


Figure 4.3: Experimental results from the CBR method using two different case databases.

method, that change the intended meaning of the sentence. Some of the sentences where this situation was observed were actually nonsense replacements or odd semantic constructs.

We also observed several syntactical errors, also present in the base problem's sentence that could be solved by enhancing the grammatical realizer as discussed earlier.

After considering the difficulty posed by some of the problems presented to the system, and although the system takes some liberties on changing the intended meaning of the output, some of the proposed solutions are actually syntactically correct and semantically coherent.

4.4 Evaluation of the Probabilistic Database N-gram Size

In our initial experiments we wanted to determine the impact of increasing the size of the N-grams database. To confirm our hypothesis that incrementing the size of the N-grams we would get significant better results we wrote an experiment to assess the performance of different databases. Inspired by the usage of N-grams by Google Scribe[34] we tried to generate synthetic text increasing the size of the

Table 4.5: Selected problems and solutions (1).

Original sentence	Octavion beamed and invigorated his handshake, explaining to Ales his immense gratitude.
Replacement	Bird beamed and invigorated his handshake, explaining to Ales his immense gratitude.
Search solution	A bird beams and invigorates his a handshake, explains to Ales his big (vs. little) a gratitude.
Observation	The replacement makes this problem especially difficult but SB handles a relatively coherent result. There is a fault in the Wordnet knowledge database that is reporting details on the synonym database.
Original sentence	Ales gripped the aristobot 's hand confidently and smiled.
Replacement	Ales gripped bird 's hand confidently and smiled.
Search solution	He gripped a bird 's a hand confidently and smiled.
Observation	This is a typical problem where the realizer correctly places a determiner but the system fails to evaluate the morphological characteristics of the actor in the predicate. This, however could be due the natural occurrence of anthropomorphic characters in common narrative.

N-gram database.

The following paragraphs of text have been generated using 2, 3 and 4 gram databases. Only a few lines are shown.

2-gram

the same time. the first time of there is the other hand. in the most of and they are not between they were not only at there are there was as the world. it is and all them. in and in this is not to the two years. he was the new york: the time to be and to and that the united states. this was not as well as to have been about the following the whole of his own. and it was to make a few days. but the second. they had been the end of this time and history of their owner of these are into the last night. and then the great deal with the state of and it. for the case of all of its owned by the old many of them to does not the people. if the fact that he had according to his wife. we have after the way of which the right to seems to take the next day. there were the only the best of our ownership of her..

Table 4.6: Selected problems and solutions (2).

Original sentence	Ales cast his eyes downward and avoided grubby workbot 's hand, unsure of how to interact.
Replacement	Ales cast his eyes downward and avoided theft incident. , unsure of how to interact.
Search solution	He cast his some eyes downward and avoided a theft an incident., unsure of how to interact.
Observation	Another phenomenon observed is the appearance of excesive determinants and undesired noun-to-pronoun replacements.
Original sentence	Ales cast his eyes downward and avoided grubby workbot 's hand, unsure of how to interact.
Replacement	Ales cast his eyes downward and avoided theft incident. , unsure of how to interact.
CBR solution	Ales cast his eyes downward and avoided theft incident. , unsure of how to interact.
Observation	The CBR approach has been observed to be rather conservative in most of the cases, defaulting to leaving the replaced sentence as is, without any further modification.

3-gram

the united states. there is not at all. there are not the only one of the world. the first time in the same time. there was able to do with the same as they are not to be able to make a good deal of the most important of these two groups of people who are not in the case of the united nations. they are the most parts of the same way as to the same. they were not the same thing. there were no longer be allowed to be the same year. the most commonly used in the united kingdom. the second world war ii. the following : there is a very large number of the first of all the time of the new york: harper and row. the same manner as to be found in the first place. these are the same in the world of the state of the two men were not to the other hand there is an important role in the form of the people of the country. the only way to them...

4-gram

the end of the year. there is no doubt that they were not to be found in the fact that they are not the same as the one who is not at all. there is also an important factor in the

Table 4.7: N-gram validation examples.

Not all results are shown.

N-gram	Length	Text	Web Results	All Results
2	6	the first time of there is	0	0
3	6	the same time of a few	3	9
3	6	the first time they can do	10	1970000000

development of the country. there is also the case that there is a great deal of time and space. there is an important factor. there is nothing to be done. there is one thing to be done in the past. there is little doubt that there is no reason to believe that there is an increase in the number of the most important of all the others. there is the same as that of the other. there is some evidence that they are the most important thing in the world. there is not a matter of fact there is no evidence that they were the first to be able to do so. there is another way of saying that there is nothing in the world of their own. there is something in the nature of the case. there is much to be desired. there is only one of the most interesting of these is the fact that it is not able to do this. there is evidence that there is not much to be said for the view that there is something to be said...

Although most of the text is nonsense, as we increase the size of the N-grams in the database, the output is greatly enhanced. We used an automated tool to look up complete sentences using an online search engine. For this experiment we define a complete sentence as a chain of words delimited by a punctuation mark. We used Microsoft's Bing API to count the number of search results for each sentence as an online query. Table 4.7 shows a couple of examples all featuring a length of 6 words.

The result count for the 4-gram database for the unfiltered result counts returned by the API were in the millions higher than those for the 2-gram database. Therefore we concluded that the ability to evaluate sentences would increase accordingly.

N-gram databases are not commonly available for N-grams of greater order than 3. We used a custom aggregated database of N-grams of order 4. We did not attempt using a greater order of N-grams due to the demanding computational requirements of increasing the size of the database.

Chapter 5

Conclusions

In this thesis we have explored two methods for Natural Language Generation (NLG) using text modification.

Specific related work similar to ours has not been found. Both Case-Based Reasoning (CBR) and Search Based (SB) methods have been used in the fields of computational creativity and narrative systems[35][36][37], but there is no evidence of the use of CBR and SB methods tailored to the specific task of grammatical realization using text modification or adaptation.

After analyzing work done in other NLG systems we argue that most of them either use constrained and highly annotated templates or expensive rule-based linguistic realizers[11]. Our goal was to provide a language agnostic system design that was able to work with loosely annotated templates and yield syntactically correct, semantically coherent output.

We have described several of the difficulties found when working with NLG and we have proposed two methods to solve NLG problems using text modification.

The conceptual model described has been tested to work with Riu, an analogy-based narrative system. As a result we have shown improvement in the surface form output of the Riu narrative system and how the system could greatly enhance several similar systems with a relative small effort. Our individual experiments demonstrate the potential of the techniques described in this thesis.

The experimental results show a significant improvement of the surface form representation obtained using our NLG system for replacement based text modification.

Although CBR has shown to enhance the output of several problems, the observed improvements are minimal compared to the blind replaced results. On the other hand, it has been observed that even with the relative simplicity of the SB method, the results outperform in about 10% the CBR method.

The main drawback of our system would be that it may sometimes change the expected meaning of the resulting modified sentence.

Currently the system explores both CBR and SB methods separately. As part of our future work, we would like to combine both methods and use CBR as a heuristic function for the SB method. The

revised output would then be used in a retain phase to enhance the CBR case database.

There are several parts that we would like research further. We plan on enhancing the grammatical realizer capabilities and expand the available Text Transformation Routines (TTR). For the retrieval engine, we would like to use dependency trees instead of strings of Part-Of-Speech (POS) particles and use approximate graph isomorphism algorithms that better match syntactic and semantic relationships between problems and cases[21].

We also want to enhance the Probabilistic Evaluator (PE) using external linguistic rule-based evaluators and probabilistic sentence frame models. We would like to study the impact of using domain-specific N-gram databases to tailor specific needs and the automatic creation of such databases from existing corpora.

Finally, our goal is to test the system design in more general NLG contexts and enable the use of the system in specific domains and languages other than English.

Bibliography

- [1] I. Encyclopædia Britannica, “Britannica concise encyclopedia,” 1994–2008. [Online]. Available: <http://www.britannica.com>
- [2] A. M. Turing, “Computing machinery and intelligence,” *Mind*, vol. 59, no. 236, p. 28, October 1950, published by: Oxford University Press on behalf of the Mind Association. [Online]. Available: <http://www.jstor.org/stable/2251299>
- [3] M. M. et al, *The Penn Treebank Project*, University of Pennsylvania, 2011. [Online]. Available: <http://www.cis.upenn.edu/treebank/>
- [4] M. Haspelmath, *Word classes/parts of speech*, 2001, vol. International Encyclopedia of the Social and Behavioral Sciences.
- [5] J. M. Ponte and W. B. Croft, *A Language Modeling Approach to Information Retrieval*, 1998, vol. Research and Development in Information Retrieval, pp. 275–281.
- [6] C. D. Manning and H. Schuetze, *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.
- [7] M. Elhadad and J. Robin, “An overview of surge: a reusable comprehensive syntactic realization component,” Tech. Rep., 1996.
- [8] M. R. Williams, *A History of Computing Technology, 2nd Edition*, 2nd ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 1997.
- [9] E. H. Hovy and Y. Arens, *Automatic Generation of Formatted Text*, 1998, vol. Readings in intelligent user interfaces.
- [10] E. Reiter and R. Dale, *Building Natural Language Generation Systems*, 2000.
- [11] W. Mann, *Overview of the penman text generation system*, University of Southern California, Marina Del Rey (USA). Information Sciences Inst., 1983.
- [12] E. Brill, “Processing natural language without natural language processing,” in *Proceedings of the 4th international conference on Computational linguistics and intelligent text processing*, ser. CILing’03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 360–369.

- [13] A. Aamodt and E. Plaza, "Case-based reasoning; foundational issues, methodological variations, and system approaches," *AI COMMUNICATIONS*, vol. 7, no. 1, pp. 39–59, 1994.
- [14] J. A. Recio-garcía, B. Díaz-agudo, and P. A. González-calero, "Textual cbr in jcolibri: From retrieval to reuse," 2007.
- [15] M. Stevenson and M. A. Greenwood, "Comparing information extraction pattern models," in *In Proceedings of the Information Extraction Beyond The Document Workshop (COLING/ACL 2006)*, 2006.
- [16] D. Rusu, L. Dali, B. Fortuna, M. Grobelnik, and D. Mladenic, "Triplet extraction from sentences," *Proceedings of the 10th International Multiconference Information SocietyIS*, p. 8Ü12, 2007.
- [17] D. Metzler, S. Dumais, and C. Meek, "Similarity measures for short segments of text," in *European Conference on Information Retrieval*, 2007.
- [18] G. R. D. Patrick A. V. Hall, "Approximate string matching," in *Computing Surveys*, vol. 12, 1980.
- [19] E. S. Ristad and P. N. Yianilos, "Learning string-edit distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 522–532, May 1998.
- [20] V. Rus, P. M. McCarthy, M. C. Lintean, D. S. McNamara, and A. C. Graesser, "Paraphrase identification with lexico-syntactic graph subsumption." in *FLAIRS Conference*, D. Wilson and H. C. Lane, Eds. AAAI Press, 2008, pp. 201–206.
- [21] M. T. Pazienza and M. Pennacchiotti, "Textual entailment as syntactic graph distance: a rule based and a svm based approach," in *In Proceedings PASCAL RTE challenge*, 2005, pp. 528–535.
- [22] A. D. Haghighi, A. Y Ng, and C. D. Manning, "Robust textual inference via graph matching," *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing HLT 05*, no. October, pp. 387–394, 2005.
- [23] H. J. Department and H. Jing, "Usage of wordnet in natural language generation," in *University of Montreal*, 1998, pp. 128–134.
- [24] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, T. G. B. Team, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. A. Nowak, and E. L. Aiden, "Quantitative analysis of culture using millions of digitized books," *Science*, vol. 331, no. 6014, pp. 176–182, 2011.
- [25] B. M. Marie-Catherine de Marneffe and C. D. Manning, "Generating typed dependency parses from phrase structure parses," vol. LREC.
- [26] H. Schmid, "Probabilistic part-of-speech tagging using decision trees," 1994. [Online]. Available: <http://www.ims.uni-stuttgart.de/ftp/pub/corpora/tree-tagger1.pdf>
- [27] P. University, "Wordnet," Princeton University, 2010. [Online]. Available: <http://wordnet.princeton.edu>

- [28] J. M. F. de Alba and J. Pavón, "Recognition and interpretation on talking agents," in *Proceedings of the 23rd international conference on Industrial engineering and other applications of applied intelligent systems - Volume Part I*, ser. IEA/AIE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 448–457.
- [29] D. Ferrucci, "Build watson: an overview of deepqa for the jeopardy! challenge," in *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, ser. PACT '10. New York, NY, USA: ACM, 2010, pp. 1–2.
- [30] M. catherine De Marneffe and C. D. Manning, *Stanford typed dependencies manual*, 2008.
- [31] N. Chomsky, *Syntactic Structures*. Mouton, 1957.
- [32] J. Zhu and S. O. nón, "Story representation in analogy-based story generation in riu," 2010.
- [33] C. Martindale, *Romantic progression: the psychology of literary history*. Hemisphere Pub. Corp., 1975.
- [34] "Google scribe." [Online]. Available: <http://scribe.googlelabs.com/static/help.html>
- [35] M. Riedl, "Story planning: Creativity through exploration, retrieval, and analogical transformation," *Minds and Machines*, vol. 20, pp. 589–614, 2010.
- [36] P. Gervás, B. Díaz-Agudo, F. Peinado, and R. Hervás, "Story plot generation based on cbr," in *Applications and Innovations in Intelligent Systems XII*, A. Macintosh, R. Ellis, and T. Allen, Eds. Springer London, 2005, pp. 33–46.
- [37] J. Kolodner, "Understanding creativity: A case-based approach," in *Topics in Case-Based Reasoning*, ser. Lecture Notes in Computer Science, S. Wess, K.-D. Althoff, and M. Richter, Eds. Springer Berlin - Heidelberg, 1994, vol. 837, pp. 1–20.