Case Retrieval Nets for Heuristic Lexicalization in Natural Language Generation

Raquel Hervás and Pablo Gervás

Departamento de Sistemas Informáticos y Programación
Universidad Complutense de Madrid, Spain
raquelhb@fdi.ucm.es,pgervas@sip.ucm.es

Abstract. In this paper we discuss the use of Case Retrieval Nets, a particular memory model for implementing case-base reasoning solutions, for implementing a heuristic lexicalisation module within a natural language generation application. We describe a text generator for fairy tales implemented using a generic architecture, and we present examples of how the Case Retrieval Net solves the Lexicalization task.

1 Introduction

Natural Language Generation (NLG) is divided into various specific tasks [1], each one of them operating at a different level of linguistic representation (discourse, semantics, lexical,...). NLG can be applied in domains where communication goals and features of generated texts are diverse, from transcription into natural language of numerical contents [2] to literary texts generation [3].

Each kind of NLG application may need a different division into modules [4]. Given a specific organization (or *architecture*) of the system, it may occur that diverse classes of application require different solutions when facing each of the specific tasks involved in the generation process. For a particular task, in processes where a quick answer is required (for instance, in interactive communication between user and machine in real time) it can be useful to use simple solutions based on heuristics, that provide quick answers even if the achieved quality is poor. On the other hand, in situations where long texts of high quality are needed with no constraints on response time it would be better to draw on knowledge-based techniques that exhaustively consider more possibilities.

The present paper proposes a case-based approach to decide which words should be used to pick out or describe particular domain concepts or entities in the generated text. The idea is that people do not create new words each time they need to express an idea not used before, but rather they appeal to the lexicon they have acquired throughout time looking for the best way to express the new idea, always taking into account existing relations between the elements of the lexicon they already know.

The paper starts with a revision of the Case-Based Reasoning and Lexicalization fields. Then we expose the fairy tale text generator where the work presented in this paper is implemented, and we consider the performance of the CBR module. Finally, the obtained results and future research lines are discussed.

2 Lexicalisation and Case Based Reasoning

Lexicalisation is the process of deciding which specific words and phrases should be chosen to express the domain concepts and relations which appear in the messages [1]. The most common model of lexicalisation is one where the lexicalisation module converts an input graph whose primitives are domain concepts and relations into an output graph whose primitives are words and syntactic relations. Lexicalisation researchers have developed powerful graph-rewriting algorithms which use general "dictionaries" that relate domain primitives and linguistic primitives. Graph-rewriting lexical choice is most useful in multilingual generation, when the same conceptual content must be expressed in different languages. The technique handles quite naturally some kinds of lexical divergences between languages. This scheme can be valid for most applications where the domain is restricted enough in order that direct correspondence between the content and the words to express it is not a disadvantage. In general, thinking on more expressive and versatile generators, this model requires some improvement. Cahill [5] differentiates between "lexicalization" and "lexical choice". The first term is taken to indicate a broader meaning of the conversion of something to lexical items, while the second is used in a narrower sense to mean deciding between lexical alternatives representing the same propositional content. Stede [6] proposes a more flexible way of attaching lexical items to configurations of concepts and roles, using a lexical option finder that determines the set of content words that cover pieces of the message to be expressed. These items may vary in semantic specificity and in connotation, also including synonyms and nearsynonyms. From this set, the subsequent steps of the generation process can select the most preferred subset for expressing the message.

Machine learning techniques have been shown to significantly reduce the knowledge engineering effort for building large scale natural language processing (NLP) systems: they offer an automatic means for acquiring robust solutions for a host of lexical and structural disambiguation problems. A good review of how they have been applied in the past to specifical NLP tasks is available in [7]. Among the learning algorithms that have been used successfully for natural language learning tasks are case based learning methods where the natural language system processes a text by retrieving stored examples that describe how similar texts were handled in the past. Case based approaches have been applied to stress acquisition [8], word sense disambiguation [9] and concept extraction [10], among others.

Case-based Reasoning (CBR) [11] is a problem solving paradigm that uses the specific knowledge of previously experienced problem situations. Each problem is considered as a domain case, and a new problem is solved by retrieving the most similar case or cases, reusing the information and knowledge in that cases to solve the problem, revising the proposed solution, and retaining the parts of this experience likely to be useful for future problem solving. General knowledge, understood as general domain-dependent knowledge, usually plays a part in this cycle by supporting the CBR processes.

Case Retrieval Nets (CRNs) [12] are a memory model developed to improve the efficiency of the retrieval tasks of the CBR cycle. They are based on the idea that humans are able to solve problems without performing an intensive search process, but they often start from the given description, consider the neighbourhood, and extend the scope of considered objects if required.

As its name indicates, CRNs organize the case base as a net. The most fundamental item in the context of the CRNs are so-called Information Entities (IEs). These represent any basic knowledge item in the form of an attribute-value pair. A case then consist of a set of such IEs, and the case base is a net with nodes for the entities observed in the domain and additional nodes denoting the particular cases. IE nodes may be connected by similarity arcs, and a case node is reachable from its constituting IE nodes via relevance arcs. Different degrees of similarity and relevance are expressed by varying arcs weights. Given this structure, case retrieval is carried out by activating the IEs given in the query case, propagating this activation according to similarity through the net of IE nodes, and collecting the achieved activation in the associated case nodes.

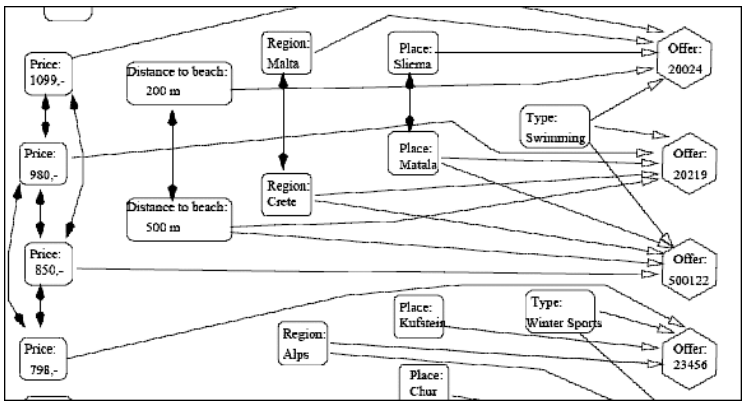

Fig. 1. An Example Case Retrieval Net in the domain of travel agencies

An example of a Case Retrieval Net applied to the domain of travel agencies is shown in Figure 1. Rectangles represent entity nodes, with their corresponding attribute-value pairs. Hexagons are case nodes, with the description that identifies them univocally. Entity nodes are related among themselves with arcs with black arrowheads, and they are related with cases by arcs with white arrowheads. Weights associated to arcs are not represented in the figure, and arcs with weight zero are ommitted.

Case Retrieval Nets present the following important features:

– CRNs can handle partially specified queries without loss of efficiency, in contrast to most case retrieval techniques that have problems with partial descriptions.

– Case retrieval can be seen as case completion. Given only a part of a case, the net can complete the rest of its content.
– The net can express different similarity and relevance values at run time by simply changing the related arc weights, while most other techniques need a new compilation to do the same.
– Cases do not need to be described by attribute vectors. There are features that would be relevant for some cases but not for the others.
– Insertion of new cases (even with new attributes) can be performed incrementally by injecting new nodes and arcs.

3 Text Generation for Fairy Tales

ProtoPropp [13] is a system for automatic story generation that reuses existing stories to produce a new story that matches a given user query. ProtoPropp receives a query from the user with the features the generated story must fulfill from the point of view of main characters, some important events that may occur, etc. The system uses this information to look for in its case base of analyzed and annotated fairy tales by means of Case-Based Reasoning (CBR). A plot plan is generated by the CBR module, structured in terms of specific instances of concepts in the ontology (characters and their attributes, scenarios where the action takes place, events that occurs,...). From this plot plan, the NLG subsystem builds the textual rendition of the story.

The specific architecture of the NLG module presented here is implemented using cFROGS [14], a framework-like library of architectural classes intended to facilitate the development of NLG applications. It is designed to provide the necessary infrastructure for developing NLG applications, minimising the implementation effort by means of schemas and generic architectural structures commonly used in this kind of systems.

cFROGS identifies three basic design decisions when designing the architecture of any NLG system: (1) what set of modules or tasks compound the system, (2) how control should flow between them, deciding the way they are connected and how the data are transferred from one to another, and (3) what data structures are used to communicate between the modules. Our solutions for these three decisions in the NLG module of ProtoPropp are the following.

3.1 Data structures

Abstract classes provided by the library allow us to create the data structures we need in our system, always taking into account that all modules must be able to work with them when necessary. A generic data structure is defined for the text that is being created, where all obtained information during system operation is stored. Any module can access all the information accumulated during the processing of the tale being treated.

Three particular inputs to the NLG module are relevant to the work described here: the knowledge base, the vocabulary and the plot plan.

The knowledge base contains all the domain information about tales the generator can consult and use, organized as a tree. This information includes characters, locations, objects, relations between them and their attributes.

The vocabulary contains all the lexical information essential to write the final text. It is structured as a tree as well, very similar to the knowledge base one, with the difference that each fact or relation has a lexical tag associated to its eventual realization in the final text.

The plot plan is the structure of the tale that is to be rendered as text. Each line of this input corresponds to a paragraph sized portion of the story, containing information about a sequence of actions, the place where they take place, the characters involved, and the objects used in them.

3.2 Set of modules

The NLG system is composed of five different modules:

- **ContentDeterminationStory.** Decides what facts of the original plot plan are going to be included in the final text. Taking into account the "historical register" of the information already mentioned we can decide what information is redundant and erase it.
- **DiscoursePlanningStory.** The discourse planning stage has to organise all the information that has been selected as relevant into a linear sequence, establishing the order in which it is going to appear in the final text. The output of this module is the rhetorical structure of the tale the module is processing.
- **ReferringExpressionStory.** This stage takes the decision of what must be the reference for each concept that appears in the tale, avoiding lexical redundancies and ambiguity.
- **LexicalizationStory.** This stage selects words from the vocabulary to describe the concepts involved in the current draft, using lexical tags for static concepts, as characters and scenarios, and templates for actions and verbs, providing structure to the sentences. Templates partly solve the need for having an explicit grammar, but the knowledge base provides the required information to solve issues like number and gender agreement.
- **SurfaceRealizationStory.** This stage is in charge of using the terms selected in the previous stage to fill in the templates. Additionally, it carries out a basic orthographic transformation of the resulting sentences. Templates are converted into strings formatted in accordance to the orthographic rules of English - sentence initial letters are capitalized, and a period is added at the end.

3.3 Flow of control information

The generation module of ProtoPropp implements the flow of control information among the modules as a simple pipeline, with all the modules in a sequence in such a way that the output of each one is the input for the next. From

the plot plan generated by the CBR module, the text generator carries out the mentioned tasks of Content Determination, Discourse Planning, Referring Expression Generation, Lexicalisation and Surface Realization, each one of them in an independent module.

4 Case Based Solutions for Lexicalisation

Lexicalisation based on templates is an acceptable method when operating in restricted domains, but results can be poor if complex actions have to be expressed. Complex actions require the introduction of lexical chains that are employed exclusively for a specific verb in some tale. This introduces an unwanted rigidity in the system, because it makes the task of extending the vocabulary an arduous one. This solution also implies that the vocabulary holds no semantic information about actors or objects involved in an action.

As an alternative, we have implemented a case-based lexicalisation module. When human beings talk or write, they do not invent new words whenever they need to express a specific idea that they have never before put into words. Instead, they search for relations between the new idea to be expressed and other ideas expressed previously, taking the same vocabulary and adapting it as required. They reuse previous experience to solve a new case.

4.1 Representation of cases

The new module operates with the same vocabulary for domain concepts and attributes as the original lexicalisation module. A lexical tag is assigned to each one of them, made up of one or more words. The vocabulary for actions or verbs becomes more complex: it is stored in the form of cases, where each case stores not only the corresponding lexical tag but also additional information concerning the type of case, the elements involved in the action, and the role that those elements play in the action.

The domain of tales is restricted, and so is the type of actions that can appear in them. The types of actions that appear in this module are:

- **Move.** Actions that result in a change of location, whether of a character or an object.
- **Atrans.** Actions where there is a transfer of possession, whether of a character or an object.
- **Fight.** Actions involving physical confrontations between characters.
- **Ingest.** Actions where a character ingests something (either another character or an object).
- **Propel.** Actions where a physical force is applied to an object.
- **State.** Actions representing a change of state of a character or an object.
- **Use.** Actions where an element of the domain participates in the action.
- **Feel.** Actions that involve feelings, both positive or negative.
- **Speak.** Actions where a character or an object express an idea out loud.

Examples of cases for the different types of action are given below:

```
FIGHT:attack, ACTOR:witch, OBJECT:Hansel
FEEL:envy, ACTOR:stepsisters, OBJECT:Cinderella, FEELING:bad
STATE:marry, ACTOR1:knight, ACTOR2:princess, INI:single, FINAL:married
```

It is important to take into account that the structure of each one of these types is not rigid. They will not always have the same elements, nor in the same order. A clear example is provided by the verbs 'leave' and 'go', both of type `Move`. The first one has an attribute `From` to indicate where the character is coming from, whereas the second one has an attribute `To` that indicates his destination.

Cases have been extracted from the formalised tales that were already available. A case is not an abstract instance of a verb or action, but rather a concrete instance in which specific characters, places and objects appear. These elements are stored in the module's knowledge base. This allows the establishment of relations between them when it comes to retrieving or reusing the cases.

4.2 The Case Base

Cases are stored in a Case Retrieval Net. This model is appropriate for the problem under consideration, because on one hand our cases consist of attribute-value pairs that are related with one another, and on the other hand the queries posed to the module will not always be complete. When the system asks for the appropriate lexical tag for a new action, the module looks for related verbs based on their type and the class of elements that are involved in them.

The vocabulary of the module is built from the case base. For each attribute-value pair in the cases an information entity is created. For each case, a node is created which holds references to the information entities that make it up. When introducing an IE, if that entity has already appeared in another case it is not duplicated. Instead, another association is created between the new case and the existing information entity.

As IEs are inserted to form the net, it is necessary to establish a measure of similarity between them. This is done by reference to the module's knowledge base, in which the different concepts of the domain are organised into a taxonomy. The similarity between two entities is calculated by taking into account the distance between them in the knowledge base and using Formula 1.

$$similarity(c1, c2) = 1 - (1 + distance(c1, c2))/10 \tag{1}$$

The distance between two concepts is calculated by finding their first shared ancestor, and adding up the distance between this ancestor and each of the concepts. It can be seen as the number of nodes we have to pass when going from one of the concepts to the other. It is also necessary to have a similarity value for each entity with itself. This value is always 1, the maximum possible.

Each of the IEs is related to the cases to which it belongs with a certain value of relevance. In the implemented module, the maximum relevance within a case

corresponds to the attribute `Type` with value 1, and the rest of the elements have relevance 0.5. This is because when retrieving cases we are mainly interested in the type of action that we are looking for, more than which elements are involved in it. However, it can occur that the module retrieves a case of a different type, if the similarity weights of the attributes of the case are high enough.

4.3 The CBR cycle

The module described in this paper executes each of the processes of the CBR cycle in the following way.

Case Retrieval The retrieval task starts with a partial or complete problem description, and ends when a matching previous case has been found. In our module, the retrieval of cases is directly handled by the Case Retrieval Net and its method of similarity propagation. Starting from a partial description of the action we need to lexicalise, formed by the information available in the input plot plan, the retrieval of the more similar cases is done by calculating an activation value for each case in the case base. The ones with higher activation are the more similar ones to the given query. This calculation is performed in three steps:

1. The IE nodes that correspond to the query are activated. If they are not in the net because they did not belong to any case in the case base, the correspondent nodes are inserted at the time of querying, calculating the similarity and relevance weights using the knowledge base. The nodes corresponding to the query are assigned an activation value of 1, and the rest a value of 0.
2. The activation is propagated according to the similarity values of the arcs. This is performed by looking over all the entity nodes of the net and by calculating for each one its activation value using its own activation and its similarity with the rest of IE nodes. This is achieved by using Formula 2 (where N is the total number of IE nodes).

$$activation(e) = \sum_{i=1}^{N}(sim(e_i, e) * activation(e_i)) \tag{2}$$

3. The achieved activations in the previous step are collected in the associated case nodes, calculating the final activations of the cases also considering the relevance weights of the arcs that connect the cases with their entities. This final activation value of the cases is calculated with Formula 3.

$$activation(c) = \sum_{i=1}^{N}(rel(e_i, c) * activation(e_i)) \tag{3}$$

Once we have the final activation in the cases, the one with the higher value is returned by the net. It would be possible to take not only the most similar one, but a set with the most similar cases to the query.

Case Reuse Each retrieved case has an associated template from the vocabulary for the verb or action it represents. In the process of reusing the case we have obtained from the net, we have to substitute the attribute values of the past case with the query values. Here we have tree different possibilities:

- If the attributes of the retrieved and the query cases are the same, the values of the past case are simply changed for the values of the query one. The template of the past case would be filled with the new values.
- If the attributes of the retrieved case are a subset of the query, then the attributes of the past case are filled with the correspondent query values, and the rest of the query attributes are ignored. The template of the past case would be filled with the new values, although we may have lost some important data of the query.
- If there are more attributes in the retrieved case than in the query, there are spaces in the corresponding template that the system does not know how to fill in. The easiest solution is to keep the values of the past case in the slots for which the query does not specify any value.

At the end of the reuse process the query has an assigned template to pass the message it is supposed to express into text.

Case Revision and Retainment When a case solution generated by the reuse task is identified as correct or incorrect, an opportunity for learning from success or failure arises. At the moment, this task is not implemented in our CBR module. Due to the constraints associated with language use, the contribution of an expert in the domain is required to revise the achieved results of the module, retaining and refining them if possible.

5 Experiments and results

The method described here has been tested over a set of formalised fairy tales that had been originally used as input data for ProtoPropp. They were constructed manually as simplified conceptual representations of existing renditions of the corresponding texts. This involved removing those sentences of the original tales with a meaning too complex to be expressed in terms of simple attribute-value pairs, such as *"The stepsisters envied Cinderella"*. The current corpus consists of five tales: the Lioness, Hansel & Gretel, Cinderella, the Merchant, and the Dragon.

To test the validity of the approach, several experiments have been carried out, using in each experiment part of the available material as case base and the remainder as test case. In each experiment, the conceptual representation of the test tale is used as input to the system, using the actions that appear in the other four formalised tales as case base. The resulting lexicalisation is then compared manually with the original lexicalisation available for the test tale. Each lexical choice is evaluated according to the scale presented in Table 1.

Table 1. Evaluation scale

Score	Observed lexicalisation	Relative Meaning
4	Matches original tale	
3	Equivalent to original	no loss of meaning
2	Valid	with slight loss of meaning
1	Acceptable	with significant loss of meaning
0	Unaceptable	radical departure from meaning

Some retrieval examples are shown in Table 2.

Results in the first example are acceptable. In the second one the meaning of the retrieved cases does not match the meaning of the query. This is due to the fact that the system stores no case for which the final state is 'alive', so it returns cases of type `State` that are not related.

The experiments are repeated for all possible partitions of the available material into pairs of test tale / sources for the necessary case base of actions. In each one two different features are evaluated: the average score of the test tale and the precision, that shows the relation between the number of retrieved cases with 3 or 4 score values and the total number of retrieved cases. The results for simple syntactic structures - presented in Table 3 - are positive, providing valid and similar cases for the queries.

6 Conclusions and Future Work

Once again, the bottleneck for this type of system lies in the process of knowlege acquisition. The use of the case-based reasoning paradigm for the task of lexicalisation is a good approximation whenever enough information is available in the case base to express in an acceptable form any new request. With a larger case base, the module would have more chances of finding a case that matches the system's need.

The main advantage of this method instead of other lexicalisation approaches is that the system does not need an exhaustive lexicon, as CRNs work by approximation and can retrieve similar cases for unknown queries due to the automatic semantic relations attained in the net.

An important point to take into account in future work is the formalisation of actions appearing in the tales. In the implemented module, the actions considered are quite simple, so they could be organised according to a simple type system. Enriching the language the system can use would slowly lead to the need for more complex actions that may be difficult to classify with simple types. For this reason, we are considering the use of primitives to build complex actions from simple ingredients. A possible starting point is Schank's Conceptual Dependency theory [15]. This has already been used for story generation in Tale-Spin [16]. Schank proposes an open set of primitives to express the meaning of any sentence. These meanings are built up from primitive using a complex

Table 2. Examples

Query	Cases retrieved	Associated template	Score
TYPE: MOVE, ACTOR: princess, FROM: castle	TYPE: MOVE, MOVE: leaveInHurry, ACTOR: Cinderella, FROM:palace	_ *left* _ *in a hurry*	3
	TYPE: MOVE, MOVE: comeout, ACTOR: Hansel, FROM: house	_ *came out of* _	3
TYPE:STATE, ACTOR: knight, OBJECT:princess, FINAL:alive	TYPE: STATE, STATE: release, ACTOR: Gretel, OBJECT: Hansel, INI:jailed, FINAL: free	_ *released* _	1
	TYPE: STATE, STATE: find, ACTOR: boatmen, OBJECT: son, INI:lost FINAL:found	_ *found* _	0

system for representing states and relationships. Some such mechanism would be very useful when extending the system to deal with complex actions. A possible solution is the use of not only attribute-value pairs, but also attribute-case pairs, where the value for some attribute may be also a whole case.

Another improvement of the module's operation would be to implement *lazy spreading activation* [17] in the Case Retrieval Net. Instead of propagating activation to all entity nodes, and then to all case nodes, propagation takes place progressively from most similar nodes to not so similar nodes. Once enough case nodes have been activated to reply to the query, propagation stops.

Table 3. Average Score for the Five Experiments

Tale	Av. Score	# Sent.	CB Size	Precision
Lioness	2.60	10	76	0.60
Hansel & Gretel	2.55	11	57	0.55
Cinderella	2.91	11	65	0.73
Dragon	3.50	6	81	1.00
Merchant	2.80	5	81	0.60

References

1. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press (2000)
2. Goldberg, E., Driedger, N., Kittredge, R.: Using natural-language processing to produce weather forecasts. IEEE Expert: Intelligent Systems and Their Applications **9** (1994) 45–53
3. Callaway, C., Lester, J.: Narrative prose generation. In: Proceedings of the 17th IJCAI, Seattle, WA (2001) 1241–1248
4. DeSmedt, K., Horacek, H., Zock, M.: Architectures for natural language generation: Problems and perspectives. In Ardoni, G., Zock, M., eds.: Trends in natural language generation: an artificial intelligence perspective. LNAI 1036. Springer Verlag (1995) 17–46
5. Cahill, L.: Lexicalisation in applied NLG systems. Technical Report ITRI-99-04 (1998)
6. Stede, M.: Lexical options in multilingual generation from a knowledge base. In Adorni, G., Zock, M., eds.: Trends in natural language generation: an artificial intelligence perspective. Number 1036. Springer-Verlag (1996) 222–237
7. Daelemans, W.: Introduction to the special issue on memory-based language processing. J. Exp. Theor. Artif. Intell. **11** (1999) 287–296
8. Daelemans, W., Gillis, S., Durieux, G.: The acquisition of stress: a data-oriented approach. Comput. Linguist. **20** (1994) 421–451
9. Ng, H.T., Lee, H.B.: Integrating multiple knowledge sources to disambiguate word sense: an exemplar-based approach. In: Proceedings of the 34th annual meeting on Association for Computational Linguistics, Morristown, NJ, USA, Association for Computational Linguistics (1996) 40–47
10. Cardie, C.: A case-based approach to knowledge acquisition for domain-specific sentence analysis. In: National Conference on Artificial Intelligence. (1993) 798–803
11. Aamodt, A., Plaza, E.: Case-based reasoning : Foundational issues, methodological variations, and system approaches (1994)
12. Lenz, M., Burkhard, H.D.: Case retrieval nets: Basic ideas and extensions. In: KI - Kunstliche Intelligenz. (1996) 227–239
13. Gervás, P., Díaz-Agudo, B., Peinado, F., Hervás, R.: Story plot generation based on CBR. In Macintosh, A., Ellis, R., Allen, T., eds.: 12th Conference on Applications and Innovations in Intelligent Systems, Cambridge, UK, Springer, WICS series (2004)
14. García, C., Hervás, R., Gervás, P.: Una arquitectura software para el desarrollo de aplicaciones de generación de lenguaje natural. Procesamiento de Lenguaje Natural **33** (2004) 111–118
15. Schank, R.: A Conceptual Dependency Representation for a Computer-Oriented Semantics. PhD thesis, University of Texas, Austin (1969)
16. Meehan, J.: Tale-spin, an interactive program that writes stories. In: Proceedings of the 5th International Joint Conference on Artificial Intelligence. (1977)
17. Lenz, M., Burkhard, H.: Case Retrieval Nets: Foundations, properties,implementation, and results. Technical report, Humboldt University, Berlin (1996)