Learning to Generate CGs from Domain Specific Sentences[1]

Lei Zhang, Yong Yu

Department of Computer Science and Engineering,
Shanghai JiaoTong University,
Shanghai, 200030, P.R.China
{zhanglei,yyu}@cs.sjtu.edu.cn

Abstract. Automatically generating Conceptual Graphs (CGs) [1] from natural language sentences is a difficult task in using CG as a semantic (knowledge) representation language for natural language information source. However, up to now only few approaches have been proposed for this task and most of them either are highly dependent on one domain or use many manually constructed generation rules. In this paper, we propose a machine-learning based approach that can be trained for different domains and requires almost no manual rules. We adopt a unique grammar system – Link Grammar [2] – for this purpose. The link structures of the grammar are more similar to conceptual graphs than traditional parse trees. Based on the link structure, through the word-conceptualization, concept-folding, link-folding and relationalization processes, we can train the system to generate conceptual graphs from domain specific sentences. An implementation system is currently under development with IBM China Research Lab.

1 Introduction

The first step of using Conceptual Graphs(CGs) in knowledge engineering, in most cases, is to construct the conceptual graphs to represent the knowledge contained in the information source. Unfortunately, automating this step regarding Natural Language(NL) source is a daunting task and many retreat to do it manually. Although manual construction suffices for some applications, it wouldn't be adequate for processing large amounts of NL data or NL data generated at run time(for example, user's query sentences). Several approaches have already been proposed for this problem, but most of them either are highly dependent on one domain or use many manually constructed generation rules. In this paper, we propose a machine-learning based approach that can be trained for different domains and requires almost no manual rules.

Conceptual graphs, when used to represent knowledge contained in natural language sentences, is serving as a semantic representation language. CGs from NL sentences, thus, is a process of semantic interpretation. Currently, this process in

[1] This work is supported by IBM China Research Laboratory.

general is infeasible for the set of all possible sentences. The method we proposed in this paper is for domain specific sentences which is the set of sentences that occur only in a specific application domain. Though the sentences are limited in one domain, our method itself is domain independent and can be trained for various domains.

Domain specific sentences are usually very stylish in the words, phrases, grammar and meaning they employ. The strong style leads to reoccurring patterns in the text for which Machine Learning techniques are applicable. Before the patterns can be learned, we utilize a unique grammar system – Link Grammar [2] – to capture the patterns through the internal syntactical and semantic link structures of the sentences. We develop word-conceptualization, concept-folding, link-folding and relationalization processes to learn to map the link structures to conceptual graphs. In short, we explore Link Grammar to map the problem of CG generation to the machine-learning area.

The paper is organized as follows. Section 2 outlines some characteristics of domain specific sentences. Section 3 introduces link grammar used in this work. Section 4 presents the detailed processes that generate conceptual graphs from domain specific sentences and shows how we translate them to the machine-learning area. Section 5 concludes our work by comparing related work.

2 Domain Specific Sentences

The set of sentences that occur only in one given application domain is called domain specific sentences. We assume that domain specific sentences have the following characteristics:

1. vocabulary set is small
2. terms and jargons of the domain appear frequently
3. grammar is stylish
4. semantic ambiguity is rare and most can be resolved by surrounding context

These characteristics inevitably exclude some domains such as novel stories which may involve a large set of vocabularies and have rich semantic ambiguity. The domains we intended are very specialized application domains such as circuit descriptions, clothes descriptions, law case documents and patients records. In these domains, the above assumptions (or characteristics) are reasonably satisfied.

These characteristics provide the basis for our work. Small vocabulary set ensures that the amount of training is limited for an adequate accuracy. Stylish grammar limits the number of possible grammar structures and enables the use of machine leaming techniques. Rare semantic ambiguity facilitates the semantic interpretation process and makes learning from surrounding context to determine the concepts and relations possible. Terms and jargons may bring some syntactic trouble. We will see how they are handled by link grammar in the next section.

3 Link Grammar

Link Grammar is a unique grammar system we employ in our work. Its link structure captures the internal structure of a sentence. We propose to use machine-learning techniques to automatically map this syntactic structure to the corresponding semantic structure – conceptual graph. We will describe the processes of mapping in section 4. In this section we will give a simple introduction to link grammar and explain why we choose it. We will use a simple sentence – I go to Shanghai – as an example. The diagram in Fig.1 is a link grammar parsing result for the sentence.

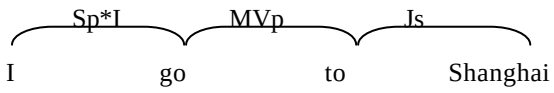


Fig. 1. The parsing result for "I go to Shanghai"

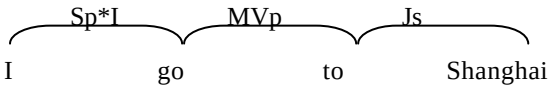
The labelled arcs between words are called *links*. Each word has a *linking requirement* stating what kind of links it can attach. The link requirements are stored in a *dictionary*. A set of links of a sentence is called a *linkage* if it satisfies the following conditions [2]: (1) Planarity: The links do not cross (when drawn above the words). (2) Connectivity: The links suffice to connect all the words together. (3) Satisfaction: The links satisfy the linking requirements of each word in the sentence. Links in Fig.1. is such a linkage. The parser of link grammar called *link parser* will find all possible linkages for a given sentence. Currently, the link parser from CMU [3] has a dictionary of about 60000 word forms and covers a wide variety of syntactic constructions. Applying link grammar to languages other than English (eg. Chinese [4]) is also possible.

Different from most other natural language grammars, link grammar's parsing result is a more flat link structure instead of a tree structure. Words that are associated semantically and syntactically are **directly** linked by the link structure [2]. In a tree structure, they may only be linked **indirectly** via their common parent nodes. This makes link structure more like a conceptual graph than a parse tree does. It can be seen clearly from the example in Fig.2. For the same sentence "I go to Shanghai", we put the link structure, parse tree and conceptual graph of it together in Fig.2 and we can make a visual comparison.

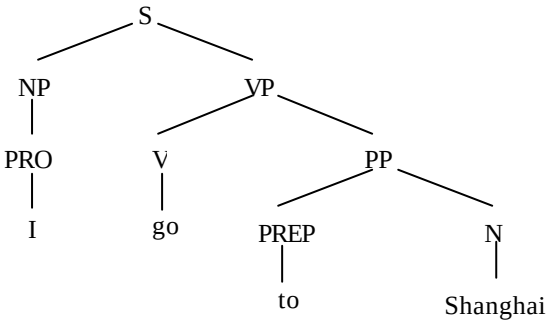
This constitutes the most important reason that we adopt link grammar in our work. It is this similarity that makes learning the mapping from link structure to conceptual graph possible. One may ask is one example enough to show that link structure is more like conceptual graph. In fact, they are similar on a deeply founded base.

Conceptual graph consists of concepts and relations. The relations denote the semantic associations between concepts. Link structure consists of words and links. The links connect syntactically and semantically related words. This isomorphism provides the basis for structure similarity. Further more, in natural language, open words (such as noun, adjective and verb) access concepts from the catalog of conceptual types, while closed words (such as prepositions) help clarify the

The link structure:



The grammar tree:



The conceptual graph:

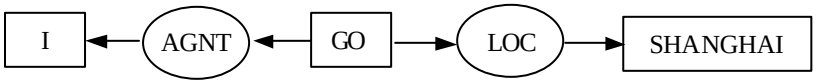


Fig. 2. Link structure is more like a conceptual graph

relationships between the concepts [5]. In the link structure, the links are also hints of the relationships between concepts. In the above example, the open word "go" represents the "GO" concept. The closed word "to" (together with its links "MVp" and "Js") between "go" and "Shanghai" represents the semantic relation "LOC". This correspondence between the link structure and the underlying semantics not only makes link structure and conceptual graph similar in structure but also makes the mapping from link structure (syntactic) to conceptual graph (semantic) possible.

Another reason that suggested us to adopt link grammar is its extensibility. The linking requirements of each word describing how it can be used in a sentence are stored in the dictionary. The grammar is distributed among words [2]. It is a lexical grammar. Thus, expressing the grammar of new words or words that have irregular usage is easy – there's a separate definition for each word. This extensibility is especially useful for the terms and jargons in the domain specific sentences that we have mentioned in section 2. What we need to do is to add or modify linking requirements in the dictionary for the terms and jargons that have irregular usage in the domain. Idioms are also handled this way.

4 Learning to generate CGs

Driven by the characteristics of domain specific sentences and the similarity between link structure and conceptual graph, we propose to automatically generate CGs by learning to map link structure to conceptual graph.

The most studied task in machine learning is inferring a function that classifies an example represented as a feature vector into one of a finite set of categories [6]. Machine learning, thus, can be seen as learning to classification. We must convert the problem of mapping to the problem of classification. We divide the mapping process into five steps and in each step a special kind of map operation is performed. The operation maps part of the syntactic structure into its corresponding semantic representation according its context. We translate the map operations into classifications of machine-learning area by encoding the operations as finite categories and encoding the contexts in which operations are performed as feature vectors. Thus, performing operations according to context is encoded and translated to classification according to feature vector.

At the same time, we propose to separate operation from its encoding in the training phase. Once the training operations have been recorded, feature vectors and categories can be encoded from them separately. This separation allows us to decide separately what constitutes context information and is encoded into the feature vector without interfering the training.

The first step of the mapping is called "word-conceptualization". This step maps words to their corresponding concepts and create concept nodes in the conceptual graph. The second step picks out concept modifiers and fold them into the existing concepts and is called "concept-folding". In the third step – "link-folding", closed words (such as prepositions) with their links that represent relations between concepts are identified and the corresponding relations are created in the conceptual graph. For those links that also represent semantic relationships, the relations are created in the conceptual graph in the "relationalization" step – the fourth step. To prevent them from interfering each other, this four steps are trained separately. In the last step, simple coreference detection and nested graph creation operations are performed. Currently, this is the only step that uses manually constructed heuristics and needs no training and learning.

In the following sub-sections through 4.1 to 4.5, we will explain the above five steps in detail and use the sentence "The polo with an edge is refined enough for work" as an example. The link structure of the sentence is shown in Fig.3. This sentence is excerpted from the corpora in our ongoing project with IBM China Research Lab. The corpora is a collection of clothes descriptions from many clothes shops on the Web.

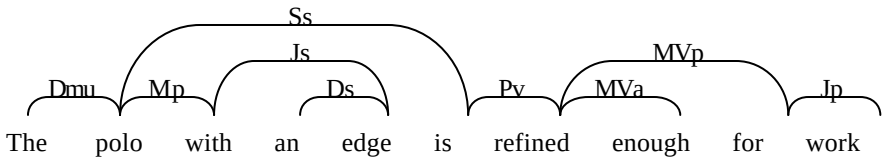


Fig. 3. The link structure for the example sentence

4.1 Word Conceptualization

In natural language, open words access concept types from domain ontology and schemata. We start the mapping from creating concept nodes for the open words in the sentence. In order to convert this concept creation operation into a classification operation, the concept type ID from domain ontology or schemata is encoded as the category name and the context information (such as part-of-speech tag, links, etc.) of the word is encoded as the feature vector. Although we have proposed to separate operation from its encoding, for convenience we will present our encoding together with the operation as an example. Its function stops there as only an example. We are not trying to make the encoding perfect in this paper.

The category for the word-conceptualization operation is encoded as the concept type ID in domain ontology or schemata. We use WordNet [7] as an experiment ontology in our project and the concept type ID is something like "WN16-2-330153" which can be used later as a key to retrieve a concept (word sense in the WordNet terminology) from the WordNet database.

For an open word *W* in a link structure, as shown in Fig.4, we encode the following context information of the operation into the feature vector:(1) word *W* in its original form (2) part-of-speech (POS) tag of *W* (3) label of the *W*'s innermost left link (4) label of the *W*'s outermost left link (5) label of the *W*'s innermost right link (6) label of the *W*'s outermost right link (7) L1's concept type ID (8) L2's concept type ID. If the links or words for the above context information do not exist for *W*, we encode *nil* in their places and we treat non-exist links as outermost links.

Fig. 4. Word *W* in link structure

For example, in the sample sentence, the context for word "refine" can be encoded as vector: < refine, VBN, Pv, nil, MVa, MVp, nil, nil >. See Fig.5. for reference. The TreeBank [8] POS tag "VBN "comes from a separate POS tagger used in our project.

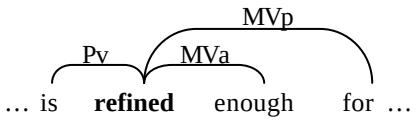


Fig. 5. The context of word "refine" in the example sentence

After this step, all concept nodes of the conceptual graph should be created. For the example sentence, the conceptual graph after this step is shown in Fig.6. For convenience, we use simple concept names in Fig.6. The concept type S-WORK is the "SUITABLE-FOR-WORK" concept type defined in our schemata which contains a variable. We will explain it in the last step: integration.

Fig. 6. Conceptul graph of the example sentence after word-conceptualization

4.2 Concept Folding

In standard conceptual graph, generic concepts are bounded by an implicit existential quantifier[1, p.86]. It is realized in the indefinite article "a" in English. In the classic example "a cat sits on a mat" in Sowa's book, "a cat" is translated into the generic concept [CAT]. This is basically what we call "Concept Folding". The indefinite article – a – is "folded" into the concept [CAT].

In order to simplify the mapping process, we extend the Conceptual Graph with many other concept modifiers in addition to the existential quantifier. These modifiers can be later removed in a equivalent standard conceptual graph. The concept modifiers include generalized quantifiers, modal modifiers and tense modifiers. Several modifiers can be combined together to modify a concept.

Generalized quantifiers are used to modify concepts that come from nouns. They include 'a' quantifier (the original existential quantifier), 'the' quantifier, 'few/little' quantifier, 'some' quantifier, 'any' quantifier, etc. Modal quantifiers are used to modify concepts that come from verbs. They include 'must' modifier, 'may' modifier, 'can' modifier, 'need' modifier, 'dare' modifier, etc. Tense modifiers also are used to modify concepts that come from verbs. They include 'past' modifier, 'future' modifier, 'progressive' modifier, 'perfect' modifier, etc. The modifying word must be a closed word. For those links that doesn't connect both a closed word and an open word, this operation is not performed in training and generating phases.

The general form of a concept and its modifier in a link structure is shown in Fig.8. A modifying word W_m is directly linked to the concept word W_c it modifies. The name of the modifier's type is used to encode the category for the "folding" operation. Because this is a simple operation, we only encode the following context information of the operation into the feature vector: (1) word W_m in its original form (2) label of the direct link between W_m and W_c.

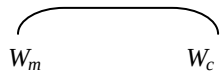


Fig. 8. Concept modifier in link structure

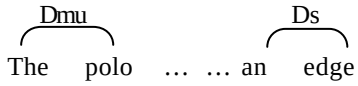


Fig. 7. Concept modifer in the example sentence

As an example, in the sample sentence, two concept-folding operations are needed. One for "the polo" and one for "an edge". The feature vectors for them are respectively < the, Dmu > and < a, Ds >. See Fig.7 for reference. Fig.9 is the conceptual graph of the sample sentence after this step.

Fig. 9. Conceptul graph of the example sentence after concept-folding

4.3 Link Folding

After the concepts have been created in the previous two steps, we are going to determine the semantic relations between the concepts. Closed words (especially prepositions) with their links indicate relationships between the concepts of the words they connect. For example, in the sample sentence, "… polo --- *with* --- edge …" shows a PART relation between [POLO:#] and [EDGE] concepts of the respective words 'polo' and 'edge'. We can "fold" the 'with' and its left and right links and replace them with a PART relation. This is called the link-folding operation.

Fig.10 depicts the general form of the link strucutre on which the link-folding operation can be performed. *W* is the closed word and it has two links connecting two other words T_1 and T_2 in the sentence. We encode the category for this operation as the type ID of the relation it creates. The relation type ID is from the same domain ontology or schemata as the concept type ID used in word-conceptualization. The feature vector is encoded from the following context information of this operation: (1) word *W* in its original form (2) POS tag of *W* (3) concept type ID of T_1 (4) POS tag of T_1 (5) concept type ID of T_2 (6) POS tag of T_2 (7) label of the link between T_1 and *W* (8) label of the link between T_2 and *W.* (As to the direction of the created relation and

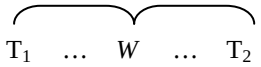


Fig. 10. Closed word *W* in link structure for link-folding

the order of the features in the vector, we can easily handle them with trivial techniques that we will not treat here)

For instance, in our sample sentence, there are three closed words that needs to be "link-folded": 'with', 'is' and 'for'. They are shown in bold italic font in Fig.11. Among them, the word 'is' is an auxiliary verb and 'with' and 'for' are prepositions. In order to distinguish more clearly the links they attach, the links in the middle that 'is' attaches is drawn in dash lines. Other links in the link structure are not drawn here.

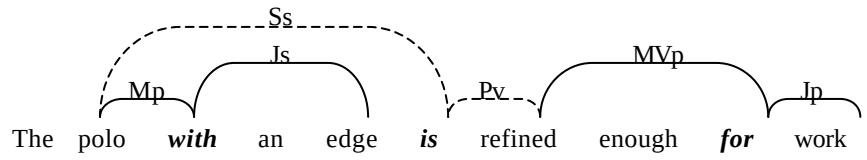


Fig. 11. The three closed words with their links in the example sentence

We still take "… polo --- *with* --- edge …" as an example. The category is encoded as the relation the operation creates: PART. The feature vector of this operation is encoded as < with, IN, POLO, NN, EDGE, NN, Mp, Js >. The relation indicated by the auxiliary verb 'is' is THEME and the 'for' between 'refined' and 'work' represents a RESULT relation.

The conceptual graph after this step has relations added between concepts. As to our example sentence, the conceptual graph has grown to Fig.12.

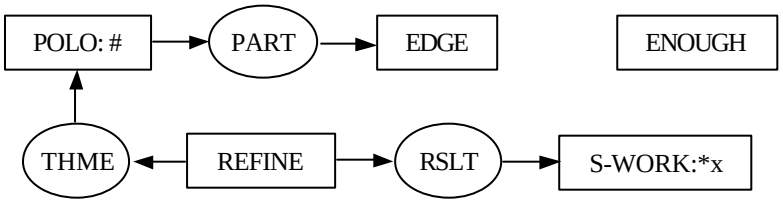


Fig. 12. Conceptual graph of the example sentence after link-folding

4.4 Relationalization

Semantic relation between concepts can also be expressed directly by a link in the link structure between the words that represent the concepts. This is the case for the 'MVa' link between 'refined' and 'enough' in the link structure of our example sentence. We can directly translate the link into a semantic relation and we call this operation "relationalization".

Fig.13 gives the general form of the link structure on which a relationalization operation can be performed. W_1 and W_2 are two words directly connected by a link that indicates a semantic relation between the concepts of W_1 and W_2. The category of the operation is encoded as the relation type ID from domain ontology or schemata. The following context information is encoded into the feature vector of this operation: (1) concept type ID of W_1 (2) concept type ID of W_2 (3) label of the link. (As to the direction of the created relation and the order of the features in the vector, we can easily handle them with trivial techniques that we will not treat here)

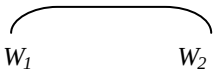


Fig. 13. The link in link structure for relationalization

In the case of our example sentence, one relationalization operation is needed for the 'MVa' link between 'refined' and 'enough'. It is shown in Fig.14. The category of the operation is encoded as the relation MANNER . The feature vector of the operation is encoded as < REFINE, ENOUGH, MVa > .

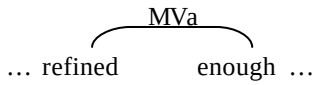


Fig. 14. The MVa link in the example sentence

After this step, more relations may be created in the conceptual graph. Regarding our example, the MANNER relation just created will connect the [ENOUGH] concept to the graph. It is shown in Fig.15.

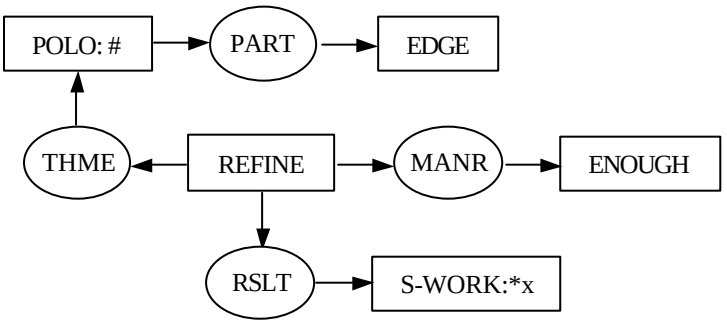


Fig. 15. Conceptual graph of the example sentence after relationalization

4.5 Integration

Integration is the last step in generating a conceptual graph. This step is not part of the training process. It only appears in the generating phase and it is the only step that uses manually constructed heuristics. What it does includes simple coreference detection and nested graph creation.

In the discussion of the previous four steps, we doesn't involve lambda expressions for simplicity. In fact, they may appear when words for concepts are missed in the sentence. They may also be introduced when concept types from domain schemata are expanded by type expansion. In order to complete the conceptual graph, we need draw coreference lines between the variables in these lambda expressions.

Although there is machine-learning based approach for coreference detection [9], in our work we mainly focus on the generation of conceptual graph for a single sentence. Discourse analysis and coreference detection is left for a separate research work. For different domains, we may construct different heuristics for them. In our current wok we only make all undetermined references to point to the topic currently under discussion.

Nested graph (context) may be introduced by type expansion or the removal of modal or tense modifiers of a concept. In our example, as we have mentioned in section 4.1, the concept type S-WORK is actually a "SUITALBE-FOR-WORK" type from the domain schemata. We can do a type expansion on it. Fig.16 is the type definition of the "SUITALBE-FOR-WORK". SUTB represents the relation SUITABLE.

type SUITABLE-FOR-WORK(x) **is**

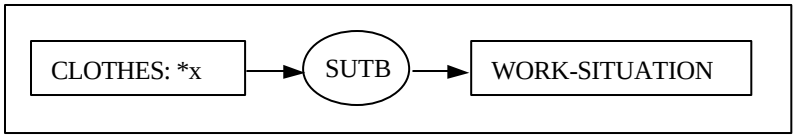


Fig. 16. The type definition for SUITABLE-FOR-WORK

After the type expansion, we can do a simple coreference detection that draws a coreference line between the undetermined variable *x* and the current topic [POLO:#]. After this step, the final conceptual graph is generated. Fig.17 is the result for our example sentence “The polo with an edge is refined enough for work”.

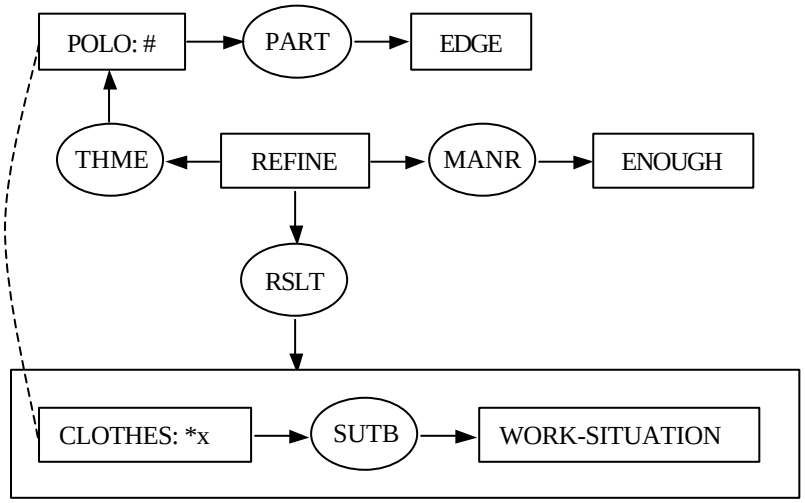


Fig. 17. The final conceptual graph of the example sentence

4.6 Summary

Through the sections from 4.1 to 4.5, we discussed the five steps in the mapping from link structure to conceptual graph and showed how we translate the mapping into machine-learning area. Word-conceptualization and concept-folding build concepts in the conceptual graph. Link-folding and relationalization connect concepts with semantic relations. For each of the four steps, we map it into the problem of classification in the machine-learning area. In the last step, we use manually constructed heuristics to do simple coreference detection and nested graph creation.

In the training phase, it is relatively easy for the trainer to determine on which words and links to perform the operations in each of the four steps, but how it is decided in the generating phase? This may be the question that one may ask in considering the whole process of conceptual graph generation. The simplest solution is to generate all possible operations and perform them. This seemingly simple solution actually works for simple domains because the number of possiblities is very low by limiting the combinations with selectional constraints and other restrictions (eg. both closed word and open word must appear in the concept-folding operation). A more elaborated solution is again to learn the knowledge. Because the number of words and links involved in each of the four operations are at most five, mapping this problem to a classification problem is trivial.

Currently a system that reads clothes descriptions from clothes shops on the Web and comprehends them in conceptual graphs is under development with IBM China Research Lab. The system is based on the ideas presented in this paper.

5 Related work

Recently there has been a significant increase in research on learning for natural language by using corpora data [6] and there are growing number of successful applications of symbolic machine learning techniques[10,11]. Applying the technique to conceptual graph generation has not yet been seen. Previous work on conceptual graph generation either use manually constructed rules or are highly dependent on one domain.

We roughly divide previous work into slot-filling and structure-mapping categoris according to their generating techniques. Slot-filling techniques such as [12,13] fill template graphs with thematic roles identified in the parse tree. Often the conceptual graph of one tree node is constructed using the conceptual graphs of its child nodes according to construction rules on how to filling the slots. This process can be done recursively bottom-up on the parse tree as in [14]. Although this approach has been successfully applied in many applications, it heavily depends on manually created construction rules on the parse tree. These rules are not only hard to create but also difficult to port to different domains. These rules mix the syntactic knowledge of the grammar used and the semantic knowledge of the domain. They create a tight coupling between the two kinds of knowledge and thus make the independent change of them difficult. In contrast, in our approach, the problem of how to perform mapping operations is translated into machine-learning problems. We need not create rules. We train the machine to learn the rules. The coupling of the syntactic knowledge of link grammar and the semantic knowledge of domain is binded by training. This coupling is loose and can be changed by training in different domains.

Another kind of technique advanced in previous work is to directly map between syntatic structure and semantic structure of CG such as [15] and [16]. We call them structure-mapping. In this respect, they are more similar to our work. To map to more flat strucutres of conceptual graphs, [15] uses syntactic predicates to represent the grammatical relations in the parse tree. Instead, in our work, link grammar is employed to directly obtain a more flat structure. Syntactic predicates are then translated to all possible semantic interpretations according to a database of translating rules. All the possibilities are finally checked against a LKB. Different from [15]'s approach, our work doesn't uses manual rules. Moreover, we separate the semantic mapping into several steps which greatly reduce the total number of possibilities. In another work in [16], parse tree is first mapped to a "syntactic CG". The "syntactic CG" is then mapped to a real CG. This approach again heavily uses manually constructed mapping rules (such as Parse-To-CG rules and SRTG rules in [16]). Further more, unlike [16]'s two-tier mapping, we do the mapping from syntactic structure in several steps but one-tier.

In [5], a multi-specialists framework for conceptual graph generation is proposed. Our approach can function as a syntax specialist in the framework. Our work presents

a preliminary inquest into the use of machine-learning technique to generate conceptual graphs from domain specific sentences. We expect that many improvements are possible and our work may be selectively adopted or enhanced.

References

1. John F. Sowa, Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, MA, 1984.
2. Daniel D.Sleator and Davy Temperley, Parsing English with a Link Grammar, in the *Third International Workshop on Parsing Technologies*, August 1993.
3. The information about the link parser from Carnegie Mellon University is avaliable at: http://link.cs.cmu.edu/link/index.html
4. Carol Liu, Towards A Link Grammar for Chinese, Submitted for publication in *Computer Processing of Chinese and Oriental Languages - the Journal of the Chinese Language Computer Society*. Abstract is available at http://bobo.link.cs.cmu.edu/grammar/liu-abstract.html
5. Graham A. Mann, Assembly of Conceptual Graphs from Natural Language by Means of Multiple Knowledge Specialists, in *Proc. ICCS'92*, LNAI 754, pp.232-275, 1992.
6. Raymond J.Mooney and Claire Cardie, Symbolic Machine Learning for Natural Language Processing, in the tutorial of *ACL'99*, 1999. Available at http://www.cs.cornell.edu/Info/People/cardie/tutorial/tutorial.html
7. George A.Miller, WordNet: An On-line Lexical Database, in the *International Journal of Lexicography*, Vol.3, No.4, 1990.
8. Mitchell P.Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz, Building a large annotated corpus of English: the Penn Treebank, *Computational Linguistics*, 19:313-330, 1993.
9. McCarthy,J., and Lehnert,W., Using Decision Trees for Coreference Resolution. In Mellish, C. (Ed.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence*, pp. 1050-1055. 1995.
10. Claire Cardie and Raymond J.Mooney, Machine learning and nautural language (introduction to special issue on natural language language learning). *Machine Learning*, 34, 5-9, 1999.
11. Brill, E. and Mooney, R.J. An overview of empirical natural language processing, *AI Magazine*, 18(4), 13-24, 1997.
12. Cyre,W.R., Armstrong J.R., and Honcharik,A.J., Generating Simulation Models from Natural Language Specifications, in *Simulation* 65:239-251, 1995.
13. Jeff Hess and Walling R.Cyre, A CG-Based Behavior Extraction System, in *Proc. ICCS'99*, LNAI 1640, pp.127-139, 1999.
14. J.F.Sowa and E.C.Way, Implementing a semantic interpreter using conceptual graphs, in *IBM Journal of Research and Development*, 30(1), pp.57-96, 1986.
15. Paola Velardi, et.,all, Conceptual Graphs for the analysis and generation of sentences, in *IBM Journal of Research and Development*, 32(2), pp.251-267, 1988.
16. Caroline Barrière, From a Children's First Dictionary to a Lexical Knowledge Base of Conceptual Graphs, Ph.D thesis, School of Computing Science, Simon Fraser University, 1997. Available at ftp://www.cs.sfu.ca/pub/cs/nl/BarrierePhD.ps.gz