Cognitive-Linguistics-Based Request Answer System

Wolf Fischer[1] and Bernhard Bauer[2]

University of Augsburg
wolf.fischer@informatik.uni-augsburg.de,
bauer@informatik.uni-augsburg.de

Abstract. Closed domains pose very specific problems to applications of all kinds. While e.g. search applications in open domains can access a huge pool of diverse information (e.g. in the internet), this is not possible for closed domains. An example application for this problem are customer support systems. These systems normally administrated by humans have to cope with different types of requests, e.g. a user asking not only a single question, but also giving a description of an experienced situation. Analyzing those complex requests is very difficult in general. Therefore it is nearly impossible to handle for common search, question-answer or customer support systems. In this paper we propose a system which should be capable of (semi-) automatically analyzing and reacting to different types of requests in closed domains based on a cognitive linguistics approach.

1 Introduction

The age of information technology has brought humanity a lot of advantages, especially regarding the accessibility of information. Besides normal key-word based search engines question-answer systems are capable of analyzing simple questions in open domains (e.g. ask.com[1], Answers.com[2], Yahoo! Answers[3] etc.). However these systems are not well suited for closed domains, because the amount of information / requests is smaller than in open domains. Following this fact there are much less syntactic variations available for similar or equivalent information. Due to the syntactic nature of normal search engines this leads to problems in answering user questions in a way which satisfies the person asking. It is therefore either up to the user to alternate his / her search query by using synonyms or an expert of the domain to answer the request.

To deal with these problems there exist different approaches. Most of todays systems focus on indexing documents with different degrees of effort. Some of them cluster documents based on syntactic information like word distribution or frequency, whereas others make use of more advanced natural language processing technologies (e.g. Answers.com) as well as certain semantic measurements

---

[1] http://www.ask.com
[2] http://www.answers.com
[3] http://answers.yahoo.com/

(e.g. with the help of WordNet). Many problems like word sense ambiguities, coreferences etc. are mainly handled by stochastic or statistical methods within generative grammar approaches.

We think that a clearly knowledge-driven approach ([7], [17]) to request-answer systems incorporating the concepts of cognitive linguistics will ease many of todays problems and deliver more satisfying results.

The novelty of our approach lies in the combination of the following points:

1. An easier way to create construction grammars for real world scenarios
2. The use of contextual data within the request analysis process
3. A self-organizing system for the analysis of text based on construction grammars, simultaneously combining syntax and common-sense as well as context semantics

The focus of this paper lays on giving an overview on our approach. It shows the first concepts and ideas behind the framework. Its implementation has started recently.

The rest of the paper is organized as follows: (2) presents some related work. Sections (3), (4) and (5) describe the ongoing project. Finally, (6) concludes the paper.

2   Related Work

Systems which can be compared to our approach are any kind of question-answer systems like the previously mentioned ask.com, Yahoo Answers etc. Most of these open domain question answer systems rely on NLP as well as document clustering technologies.

There are also many systems which rely on additional knowledge, e.g. [16], [15] or [6]. More recent systems are Wolfram Alpha[4] as well as The True Knowledge Answer Engine[5]. The latter is a new kind of search machine which seems to use a similar way to our approach. They maintain a fact driven knowledge base consisting of thousands of facts, which is altered by humans. There are however some differences between their approach and ours: First questions are being converted to a query which returns results from the knowledge base, therefore there is a direct separation between knowledge and text. Secondly the system is only capable of parsing simple questions whereas our aim is to also use the context given by a prior request.

In the field of computer science there have been different approaches to language processing using the foundations of CxG, e.g. Embodied Construction Grammars ([1]) and Fluid Construction Grammars ([13], [14], [12]). The goal of FCG is to create a linguistic formalism which can be used to evaluate how well a construction grammar approach can handle open-ended dialogues ([13]). To evaluate the approach it has been implemented within autonomous, embodied

---

[4] http://www.wolframalpha.com/
[5] http://www.trueknowledge.com/

agents. Some of the key assumptions of FCG are that it is usage-based (inventories are highly specialised), the constructions are bi-directional (i.e. FCG can handle parsing as well as production of language), it uses feature structures (which are directly incorporated within the constructions) and there is also a continuum between grammar and lexicon ([14]). The production as well as parsing process is handled by a unify and merge algorithm, which allows for an emergent creation of either the semantics or the syntax using best-match-probabilities in the unification of the constructions. This leads to a high robustness of the algorithm and more natural creation of language.

FrameNet is a project which creates a lexical resource for English based on frame semantics ([11]). Recent research tries to combine FrameNet with constructions ([5]), as certain linguistic structures can not be detected by the current mechanisms. In comparison to FrameNet, our approach a) yields for specific domains and b) tries to analyze a text with the goal to identify the users intention based on what is already known about possible requests (see 4.2 and 5.1). Still, FrameNets knowledge could serve as a good common sense basis.

Texai[6] is an Open Source approach to an artificial intelligence system. Its target is "to acquire linguistic and common sense skills that improves its own performance". Therefore it also uses a common sense knowledge base (currently based on OpenCyc) as well as FCG to handle the interpretation of natural language. The project is currently in development and therefore still lacks some knowledge, especially for the FCG component.

FCG and its concepts will serve as a foundation for our ongoing project. The project is going to be used in different types of domains therefore adjustments will have to be made, some of which will be introduced in the following sections.

3 Problem

Our main challenge is the analysis of textual requests of any kind. The system must therefore be able to identify the knowledge as well as the intention of the user as best as possible and 'act' accordingly. One of our main tasks is therefore a consequent way to combine semantic knowledge and language in order to gather the knowledge of a domain. Due to the amount of knowledge needed to make such a system reliable this approach has rarely been tried in the past. In cases this way has been tried there has been a clear separation between language (e.g. a normal glossary) and knowledge itself (e.g. a domain model) without direct combination of these two worlds. However it is clear that knowledge is needed in order to fully understand language (e.g. to disambiguate language [7], [17]). Another problem that we will try to tackle is the inclusion of additional data in the analysis process like earlier requests, context- and / or profile-information ([2],[10]). Basically each piece of information which could be relevant will be usable within the analysis process. As the system should not only be a one-way experience, feedback of as well as interaction with the user is important ([8]) and will also help to improve the accuracy of the system.

---

[6] http://www.texai.org

4 Knowledge Structure

This section describes the reference meta model (4.1) which acts as the central model to the different knowledge parts. Each part is accompanied by a simple example for demonstration. The examples notation is leaned to UML.

As seen in figure 1, the meta model is separated in different **Scope**s. Scopes are there to model different aspects of a **Domain**. A Domain would be e.g. a car manufacturer or a computer seller. Therefore a Domain contains everything that is needed in order to answer requests which are specific to that certain Domain.

As can be seen in figure 1, there are different specializations of scopes. The three main ones are the **SemanticScope**, the **SyntacticScope** and the **ConstructionScope**. The SemanticScope is further endetailed by a **SemanticProfileScope** as well as a **RequestAnswerScope**.

Scopes can be seen as some kind of view onto a database (in this case the domain) and not a clear separation. This way the syntax-lexicon continuum stays intact, as all the data is stored in the domain itself (e.g. [3], [9]).

Another argument for this separation is that we think it will be easier to let experts handle their specific fields without interfering in other experts fields. This way a classic 'domain' expert can care about modeling and populating the semantics of the domain, another one (e.g. a linguist) handles the syntactic scope and a last one finally brings both these worlds together by creating the construction scope. Especially in contrast to the way constructions are handled in FCG this should improve the time which is needed to create constructions.

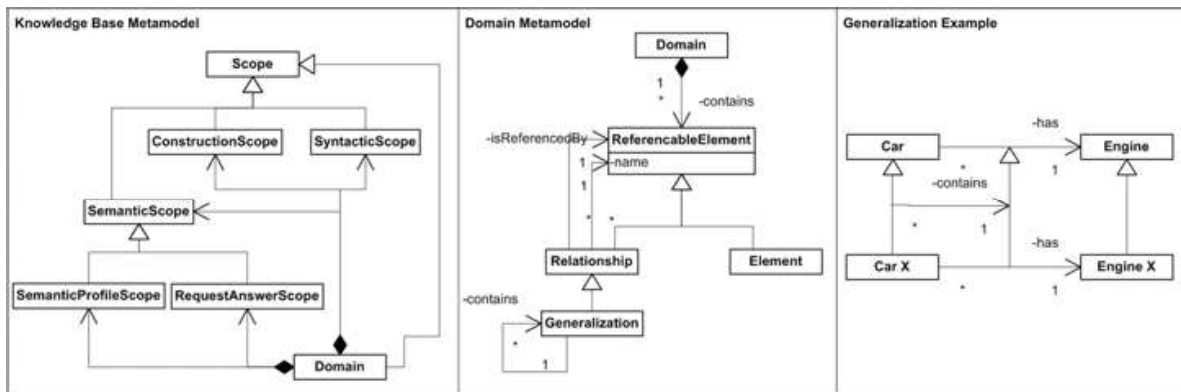The five different scopes will be described in the following paragraphs.

Fig. 1. Overview of the knowledge base structure (left), domain metamodel (middle) and example (right)

4.1 Domain

The Domain is the scope, which contains all data of the domain to be modeled. Therefore it acts as a) a container for all the data within the actual domain as well as b) a reference model for all the data which can exist within the domain. This allows the direct combination of different parts of the model, as described later in this paper.

The structure of the domain is seen in figure 1. Central to the Domain is the **ReferencableElement** (RE). Everything which should be referenced within the KB is of type ReferencableElement, starting with the **Relationship** and the **Element**. The former is used to relate REs with each other. As a Relationship is an RE itself, it can reference other REs as well as Relationships. An Element is a more concrete specialization, used to express different kinds of concepts in later phases. Central to our KB is the **Generalization**, which will be used in every Scope. It allows the creation of taxonomies in the KB. Like in modern programming languages there is be the possibility to overwrite certain parts of the hierarchy. An example is shown in figure 1. In there the SemanticElement 'Car X' specifies that it has exactly an engine 'Engine X' and not 'Engine'.

In the following sections all elements of the Domain, which will be reused are pictured in the figures using a gray background.

4.2 SemanticScope

The SemanticScope is about modeling all the factual knowledge which is inherent to the actual domain, e.g. specifying that a house consists of walls or a car has an engine. To model this kind of facts the model sticks to a generic approach. Central to the SemanticScope are the **SemanticElement** (SE), the **Generalization** as well as the **Association**. A SemanticElement is a specialization of the Domain::Element, as can be seen in figure 2. It is used to represent the concepts (in the former example 'Car' and 'Engine') which are relevant to a specific domain. An Association is used as a generic mechanism to relate SemanticElements between each other. The type of the Association is, opposite to other ontological standards, defined by a SE that the Association can reference via the 'isOfType'. This will help us later in combining syntax and semantics, using constructions.

As already mentioned there are two further specializations of the SemanticScope: The SemanticProfileScope and the RequestAnswerScope. The former will hold additional semantic information about the current user, whereas the latter contains knowledge about what is needed in order to match a request to a specific answer.
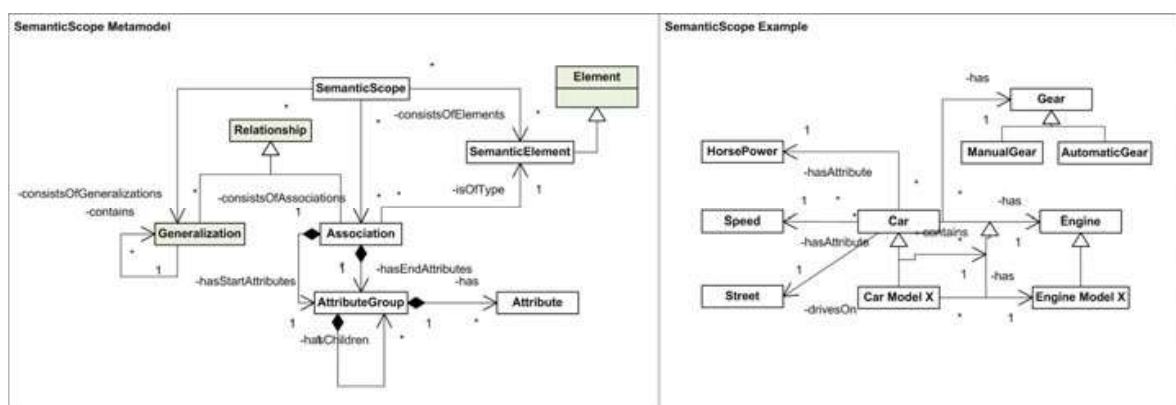

Fig. 2. Overview of the SemanticScope of the metamodel and an example

Figure 2 shows an excerpt of a SemanticScope, which contains information about cars (note that instead of an actual link to another concept, as described in the knowledge structure, the type of an association is given as the textual description of an association, this way keeping a better overview).

4.3 SyntacticScope

The SyntacticScope references all the forms which will later be used to represent the actual concepts, e.g. the SemanticElement 'Car Model X' could be referenced by the actual forms 'Car' or 'Model X'. The SyntacticScope (figure 3) contains the **SyntacticElement** at the top. This is inherited to the **Form** as well as the **SyntacticCategory**. A Form will actually represent the different strings (in the example above 'House' or 'Building'). Each Form can further be associated with a SyntacticCategory (e.g. 'Substantive', 'Verb') etc.. This is in alignment with the fact that different cultures use different syntactic categories, therefore we can include an arbitrary amount of syntactic categories. However it is a very time consuming task to add every possible form of a word (just think of verbs and their different forms throughout different times like go, went, gone etc.). Therefore there is a more generic way to represent forms, i.e. the root of a word (**FormRoot**). This will serve as the basis for a collection of different forms of the same word. A FormRoot can further directly associate its more specific full forms.

An example of a SyntacticScope is seen in figure 3. The FormRoot "Driv" is referencing three more specific occurences: "Drive", "Drives", "Driven". The first one can either be a verb (thus referencing the SyntacticCategoryGroup SynCatGroup 3) or a noun.

4.4 ConstructionScope

The ConstructionScope contains all the information necessary for combining the SemanticScope with the SyntacticScope. This is done by creating **Construction**s. A Construction contains **Symbol**s, which can be linked by **SymbolLink**s. A Symbol can either be a **SemanticSymbol** (referencing a SemanticScope::SemanticElement), a **SyntacticSymbol** (referencing a SyntacticScope::SyntacticElement) or a **Unit** (referencing several other Symbols at once).
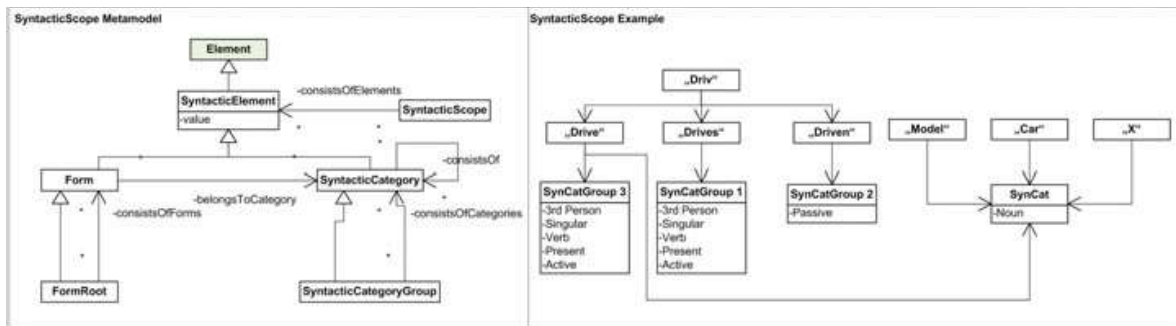

Fig. 3. Overview of the SyntacticScope of the metamodel and an example

Table 1. Construction for mapping a SemanticElement to its forms

Attribute	Content
Name:	'Car Model X' representation
Syntactic Symbols:	$Model(m), Car(c), X(x)$
Semantic Symbols:	$CarModelX(cmx)$
Relations:	$Aggregates(u, \{m, c, x\})$
Mapping:	$u \Leftrightarrow cmx$

To create a mapping between a SemanticSymbol and a SyntacticSymbol, the SymbolMapping is used.

In case of abstract constructions (i.e. constructions which mainly reference syntactic categories), there can be further specifications about the referenced (syntactic or semantic) elements. For example, there could be an abstract construction C, which contains two semantic symbols s_1, s_2. In case of both symbols referencing the same semantic element e, a SemanticRelation could state, that the textual occurrences of s_1 and s_2 must be different subtypes of e: $subtype(e, s_1) \neq subtype(e, s_2)$.

The following lines will give examples on how Constructions can associate the SyntacticScope with the SemanticScope. The construction in 1 shows how the SE 'Car Model X' is associated with its corresponding words. First, the different syntactic elements are referenced, namely the Form "Model", "Car" and "X". These three elements are the same as in figure 3. Each element is therefore associated with a variable name (standing in braces behind the corresponding element). Next, the SEs are defined, in this case the 'Car Model X' only. The construction has to specify, that all three syntactic symbols should be treated as one, therefore 'aggregating' these three into one. The new element, which will represent the three single ones, is called u and corresponds to a 'Unit' within the ConstructionScope (4.4). Finally the mapping between the SyntacticScope and the SemanticScope is specified by specifying $u \Leftrightarrow cmx$. Another more abstract

Fig. 4. Overview of the ConstructionScope of the metamodel

Table 2. Construction for a noun phrase

Attribute	Content
Name:	NounPhrase
Syntactic Symbols:	$Noun(n), Adjective(a), Determiner(d)$
Semantic Symbols:	$Element(e1), Element(e2)$
Relations:	$relation(e1, e2), inOrder(d, a, n)$
Mapping:	$n \Leftrightarrow e1, a \Leftrightarrow e2$

example is given in 2. It specifies what a noun phrase looks like. Therefore, the SyntacticElements Noun, Adjective and Determiner a referenced. Further two abstract Elements $e1$ and $e2$ are specified. Next it is defined that both SemanticElements must be related somehow (specified by $relation(e1, e2)$) and that the SyntacticElements must exist in a specific order, i.e. determiner → adjective → noun. Finally, the mappings between the $e1$, $e2$, n and a are specified. A final example will use the previously defined construction 2 for specifying a complete sentence structure. It consists of a noun phrase → verb phrase (specified accordingly to the NounPhrase construction) → noun phrase. In order to reference other constructions, a new field 'Other Constructions' is introduced (which is equivalent to the 'makesUseOf' association in figure 4). As a Construction consists of a semantic and a syntactic part we can use these parts in further specifying the relations in the new construction. There it specifies that the semantic part of noun phrase 1 and the verb phrase must be related (the same accounts for the verb phrase and noun phrase 2). Further the syntactic parts of noun phrase 1, the verb phrase and noun phrase 2 must be in order.

5 Request Analysis

To accomplish our goals we will use an iterative approach which tries to 'shape' the result until it matches users intention. Each iteration is comprised of different steps. First the user enters her request as she would in any generic online support

Table 3. Construction for a simple sentence structure

Attribute	Content
Name:	NP-VP-NP
Other Constructions:	$NounPhrase(np1), VerbPhrase(vp),$
	$NounPhrase(np2)$
Syntactic Symbols:	
Semantic Symbols:	
Relations:	$relation(np1.semantic, vp.semantic),$
	$relation(vp.semantic, np2.semantic),$
	$inOrder(np1.syntax, vp.syntax, np2.syntax)$
Mapping:	

site. Next the system analyzes the input by simultaneously analyzing the syntax as well as the semantics, which is in line with Texai, i.e. newly created knowledge is checked based on a common sense knowledge base. However as there are different foci between our project and Texai or FCG on the goals to be achieved, there are some things left which must be adapted in order to make the system usable for our intentions.

One of these things is incorporating knowledge adapted in prior contacts with the user. The system will therefore store prior requests as well as specific user information within a corresponding semantic profile. Further a semantic 'guessing' based on the RequestAnswerScope information will reduce the overall processing complexity of the system. Additionally, if not pleased with the result the user can alter his request and let the system process it again. The next paragraphs will give an outlook on the concept behind the analysis steps.

5.1 Textual Analysis

The brain is often referred to as an emergent system as it is not (yet) deducible for humans, how a large collection of neurons can yield such complex and still reasonable behavior. The emergence attribute also accounts for language parsing and production, which is sometimes also referred to as the most complex task humans are capable of. Therefore the system is going to be designed as a self-organizing system. We will first have a look at the different components of the system. On the micro level the system consists of (Construction-) Cells (CC). If being created a cell is omni-potent, i.e. it can differentiate into any known construction (it's the cells 'DNA'). Differentiation means that a cell looks for a construction which seems promising to its current local context and interprets this construction with all local information available to the cell. This allows a cell to perfectly fit into a specific context. A cell is able to fission. Depending on the fission direction (down, up or aside) the new cell has prior information helping it differentiate (only if fission is going downwards). No cell is able to control other cells (which is one aspect of a self-organizing systems ([4])). Each cell is attached to a specific level (see 5): Level 1 contains the cells which are directly attached to the text. Level S contains the cells holding construction knowledge about complete sentences. In between are cells which span a web between Level S and 1. The Interpretation Organism (IO) is the macro level of the system and therefore represents the macro behavior which is a direct implication of the micro level. The macro behavior is seen in the semantic as well as syntactic interpretation which are being created by the single cells.

The following lines will show a short scenario of how the cells can adapt to a sentence in a decentral and self-organizing manner. In order to analyze text an IO is initialized with the text (e.g. 'She drives'...) as well as a single cell (CC1). As text is expected to consist of sentences the first cell differentiates into a generic sentence cell (i.e. a sentence is just a sequence of words) and is therefore located on Level S. Based on the premise that the SyntacticCategory 'Sentence' can consist of 'Word's (e.g. in English or German), the cell fissions downwards, trying to fill the wholes in its sentence construction. The new cell
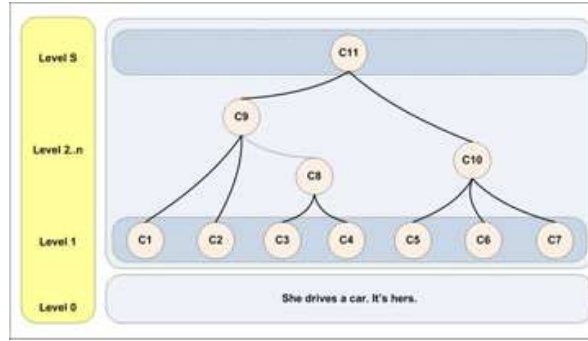
Fig. 5. Illustration of the different cell levels

(CC2) has the information about being a word from its parent and therefore differentiates based on a construction which can be attached to the first position of the text (i.e. 'She', possibly a 'Subject'). Attachment to the text means that this construction will be located on Level 1. As CC2 'sees' that there is more text left it fissions into a new cell (CC3), i.e. it fissions aside. The new cell is located on level 1 and therefore attaches to the next word (i.e. 'drives', potentially a 'Predicate'). At the current state there are now 3 cells: One on L-S and two on L-1. The new cell on L-1 knows that there is a direct neighbor but there is no direct connection to it yet i.e. there is a cell missing which connects both. CC3 is alone and therefore fissions into a level above its parent (it can and will also fission aside). The new cell (CC4) tries to attach to CC2 and CC3, as both are near to CC4. Based on the information given by these cells it searches for a construction matching on CC2 and CC3. In our example fitting constructions could be any starting with a 'Subject' + 'Predicate' description which leads to a simple 'Subject - Predicate - Object'-Sentence. Further the construction has the information that the subject should be related to the predicate. Based on the available domain knowledge the cell can confirm that a person ('She') can drive. This makes CC4 a better match for this context than CC1. Therefore CC2 dismisses its connection to CC1 making it possible for CC4 to dock to CC2 and CC3. CC1 however 'dies', therefore making room for CC4 on L-S.

The prior example should illustrate the way the algorithm in a self-organizing manner manages the creation of a construction network. As this only illustrated the basic process behind the self-organizing process, there are more mechanisms needed in order to give feedback to this dynamic aspect of the system. Most of these incorporate feedback mechanisms, especially between the micro- and the macro- levels. These will be explained in the following lines.

If a CC attaches to another CC, it will trigger an increase of importance of the referenced semantic elements (SE). This increase works in two phases:

1. Phase 1 is the direct attachment of a new CC, in turn increasing the corresponding SEs.
2. Phase 2 is the attachment of a new CC to another CC, yielding another one-time increase of the SEs importance of the 'old' CC.

Phase 2 can therefore be seen as a heightening of the context relevance of the SEs. However there can also be an increase of importance in the other direction, i.e. from the SEs to the CCs, again working in two phases:

1. Phase 1 is in case of the initial increase of importance of an SE leading to an importance increase of all CCs attached to this SE.
2. Phase 2 is in case of the SEs being part of a possible semantic request. If this request is referenced by a certain amount of SEs, it will trigger an increase of importance of all attached CCs.

This way of strengthening (especially phase 2) accomplishes our goal of 'guessing' the correct request from the start and thus also helping reducing overall processing complexity. The importance of the cells and elements directly affects the cells 'willingness' to dock to other elements.

The strengthening of importance of different elements is also the key component to easily incorporate existing knowledge of a user into the analysis process. Therefore, prior knowledge will be marked as having a high importance at the beginning.

Following these description the cells participate in the selection of an answer which should match users input best.

6   Conclusion

In this paper we have given an overview of a system which should be capable of handling all sorts of user requests in case the corresponding information is available to the system. Therefore it heavily relies on a domain dependent ontology containing information about factual as well as language related knowledge. By the strong connection of these both worlds we hope to achieve better results in the analysis of textual customer requests.

This promising approach is currently in development and will yield first results soon. Existing corpora (e.g. from the Delph-In[7] project) and databases will be reused especially for languagerelevant information.

References

1. Bergen, B., Chang, N.: Embodied construction grammar in simulation-based language understanding. In: Construction grammars: Cognitive grounding and theoretical extensions, pp. 147–190 (2005)
2. Chai, J., Jin, R.: Discourse structure for context question answering. In: Proceedings of the Workshop on Pragmatics of Question Answering at HLT-NAACL 2004, pp. 23–30 (2004)
3. Croft, W.: Radical construction grammar: Syntactic theory in typological perspective. Oxford University Press, USA (2001)
4. De Wolf, T., Holvoet, T.: Emergence versus self-organisation: Different concepts but promising when combined (2005)

---

[7] http://www.delph-in.net/

5. Fillmore, C.: Border conflicts: Framenet meets construction grammar (2008)
6. Fu, J., Xu, J., Jia, K.: Domain ontology based automatic question answering. In: International Conference on Computer Engineering and Technology, ICCET 2008, vol. 2, pp. 346–349 (2009)
7. Isahara, H.: Resource-based natural language processing. In: International Conference on Natural Language Processing and Knowledge Engineering, NLP-KE 2007, 30 August - September 1, pp. 11–12 (2007)
8. Kaufmann, E., Bernstein, A.: How useful are natural language interfaces to the semantic web for casual end-users? In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 281–294. Springer, Heidelberg (2007)
9. Langacker, R.W.: An introduction to cognitive grammar. Cognitive Science: A Multidisciplinary Journal 10(1), 1–40 (1986)
10. Lin, J., Quan, D., Sinha, V., Bakshi, K., Huynh, D., Katz, B., Karger, D.: The role of context in question answering systems. In: Conference on Human Factors in Computing Systems, pp. 1006–1007. ACM, New York (2003)
11. Ruppenhofer, J., Ellsworth, M., Petruck, M., Johnson, C., Scheffczyk, J.: Framenet ii: Extended theory and practice (2006) (unpublished manuscript)
12. Steels, L.: Fluid Construction Grammar Tutorial. Tutorial (2004)
13. Steels, L., De Beule, J.: A (very) brief introduction to fluid construction grammar. In: Proceedings of the Third Workshop on Scalable Natural Language Understanding, pp. 73–80. ACL
14. Steels, L., De Beule, J.: Unify and merge in fluid construction grammar. In: Vogt, P., Sugita, Y., Tuci, E., Nehaniv, C.L. (eds.) EELC 2006. LNCS (LNAI), vol. 4211, pp. 197–223. Springer, Heidelberg (2006)
15. Tapeh, A., Rahgozar, M.: A knowledge-based question answering system for b2c ecommerce. In: Fifth International Conference on Information Technology: New Generations, ITNG 2008, pp. 321–326 (2008)
16. Hermjakob, U., Hovy, E.H., Lin, C.-Y.: Knowledge-based question answering (2002)
17. Wang, Y.: A tri-level knowledge representation model for nlp. In: IMACS Multiconference on Computational Engineering in Systems Applications, vol. 1, pp. 547–553 (October 2006)