Ontology based Spreading Activation for NLP related Scenarios

Wolf Fischer
Programming Distributed Systems Lab
University of Augsburg
Augsburg, Germany
wolf.fischer@informatik.uni-augsburg.de

Bernhard Bauer
Programming Distributed Systems Lab
University of Augsburg
Augsburg, Germany
bauer@informatik.uni-augsburg.de

***Abstract*—To handle the flood of information in the modern world new technologies are needed. One problem is the handling and filtering of information itself. Semantic technologies have been named to possess the potential to at least facilitate this problem. Another difficulty is the representation of information to humans. Different algorithms and user interface concepts have been created allowing the access on a very specific type and structure of information. However the most common and natural way for humans is to use natural language. Natural Language Processing tries to analyze the syntax and semantics of language, but often delivers unsatisfying results because of the many phenomena (e.g., ambiguity) humans use while communicating. We therefore currently develop an approach, which allows us to analyze the semantic content of natural language text based on an ontology. In this paper we present a spreading activation based algorithm, which not only helps identify the correct semantic concepts for a natural language text, but also partially solves other phenomena of natural language.**

***Keywords*-semantic. spreading activation. natural language. ontology.**

I. Introduction

Ontologies have provided a comfortable way to store and perform reasoning on semantic information. Different standards have been proposed in the past of which OWL ([1], [2]) became the de facto standard. The availability of standards lead to ontologies being used even in big companies (e.g., the automotive sector). However this introduced a new problem as a new source of information is stored independently from all the other existing information. This is especially a problem for natural language documents, which contain the same or similar information as domain specific ontologies.

Currently there are no concepts or components available to bridge this gap, i.e., the gap between semantic and syntactic information (we refer to syntactic information as meaning both lexical and syntactic information). Ontologies only contain semantic information, but lack the syntactic part. On the other side documents contain a lot of information, which is stored in natural language form. Todays natural language processing components are capable of analyzing the syntactic information with a certain degree of precision. However this still leaves the question how semantic information can be gathered from the documents and how this information can be mapped to an ontology.

At the moment we are developing a prototype, which creates a model, which links a given text to an ontology (i.e. it does not extract new information from text, but try to link different types of information, i.e., natural language documents and ontologies). However there are many challenges because of the different types of ambiguities humans tend to use while writing. Many of those problems can only be solved during runtime i.e., during the analysis process. For example identifying the correct concept for a given form requires context and background knowledge. In our case this knowledge exists within an ontology. The 'easiest' case is if one word is mapped to several different concepts and one of these concepts is the correct one (e.g., 'bank' might mean the financial institute as well as a physical object, which is used for sitting). However in other cases humans tend to either use more abstract forms for what they actually mean (e.g., they refer to just 'the car' however they refer to their very own type of car). Also they could use a word, which has nothing to do with what they actually mean (e.g., in a sentence like 'I drive a red one', 'red' can be an indication towards a specific car, which is colored red). As can be seen by those simple examples there are many different cases in which it is not trivial to identify the correct meaning of a word.

We are currently developing a concept, which tries to solve this problem. A consistent meta model, which combines semantic and syntactic information at an early stage has already been presented ([3], [4]). We have developed an algorithm based on spreading activation (i.e., a marker distribution within a graph like data structure), which helps us solving exactly those problems as mentioned before. The algorithm itself is not a complete Word Sense Disambiguation (WSD) algorithm, but represents a core part of it as our WSD is based on calculating the semantic relatedness between concepts. Some more details are given in Section III.

This paper is structured as follows: Section II presents related work. Next, Section III gives a short introduction in our previous work, on which this algorithm is based on. Section IV specifies the requirements our concept has to fulfill. In Section V the concept is presented, before in Section VI several examples demonstrate the working mechanism of the algorithm. The paper is concluded in Section VII.

II. Related Work

Spreading Activation is a famous approach in many different areas, e.g., in cognitive linguistics as well as WSD. The latter is closely related to our problem (as mentioned in the introduction), therefore we will especially delimit our concept from WSD approaches.

The most closely related concept to our approach seems to be that of Kleb and Abecker ([5]), which disambiguate word senses based on RDF graphs. They state homonymy and synonymy as their main problems (whereas we differenciate some more problems as stated in the introduction). Their approach does however not directly regard the problem of overgeneralization as well as words, which reference a seemingly unrelated concept at first.

Tsatsaronis et al. ([6], [7]) describe a spreading activation based approach, which uses the information from a thesauri to create a spreading activation network (SAN) for WSD. Their concept is used to disambiguate complete sentences at once. The background knowledge used is from WordNet 2. Their approach is not capable of 'guessing' better suited concepts than those, which have already been found. In ([8]) Tstsaronis et al. further evaluate the state of the art of using spreading activation for WSD. They state that concepts, which use semantic networks show the best results.

Other approaches to WSD are seen by Agirre et al. ([9]), which use a PageRank based algorithm to disambiguate word senses in the biomedical domain. Kang et al. [10] created a semi-automatic, domain independent approach to WSD (whereas we focus on specific domains). An ontology is created semi-automatically and then used for disambiguating the words of a given sentence by finding a least weighted path through the concepts within the ontology. In contrast to our approach they seem to be limited regarding the identification of the correct sense for seemingly not related words (e.g., 'red' can still refer to 'car') as they rely on WordNet only.

Spreading Activation has been used in other domains as well. Hussein et al. ([11]) used it for context adaptation. Therefore they model application domains within an ontology and after each user action an activation flow through the network filters those nodes, which are seemingly most important to the current circumstances.

III. Basics

Our approach is based on a consistent meta model combining semantic with syntactic information ([3], [4]). Our algorithm uses the semantic information available and automatically identifies the most probable concepts at hand. Based on this, syntactic structures can be mapped to specific semantic structures.

Our prototype gets as an input a natural language text, which is first being preprocessed (i.e., tokenized, POS tagged and then a syntax tree is being created). Afterwards this information is used to parse the syntax tree bottom-up and create new semantic information based on previous information. To check how existing information can be combined the algorithm takes the ontology into consideration. The focus of this paper is exactly on that step of the analysis. The algorithm is called with two or more concepts and returns a value indicating the semantic relatedness. Further it might determine concepts, which might be better suited based on the context of the original input concepts. These new concepts will then be integrated into the solution set. The best concepts with respect to a global solution are then selected as part of an evaluation in the following steps. The final result is a semantic model of the initial input text, i.e., it contains, which words of the text correspond to which concept in the ontology. Further the relations of the concepts as indicated by the text are stored in the semantic interpretation result.

The algorithm in this paper is therefore a key component within our overall analysis process and has a great influence on the outcome of the result. Its working mechanism is described in the following sections.

IV. Requirements

As mentioned previously there are several cases, in which it is difficult to identify what concepts a human might have related to. The following gives a short overview of the requirements our approach has to fulfill.

1) Analyzing text requires disambiguating the senses of each word. Therefore it is necessary to have some kind of measurement indicating if different concepts are semantically related to each other. We assume that this information helps us in solving the WSD problem. Therefore the algorithm should return a value between 0 and 1, which indicates if specific information is available within the ontology and how closely it is related. 1 should indicate that there definitely is such a relation available. 0 means that no information could be found. This is important for disambiguating synonyms and homonyms in general.
2) As humans tend to overgeneralize their expressions (e.g., instead of talking about 'E3' in Figure 1 they talk of their 'Car') our concept should be capable of identifying the most specific information possible (hyponym), i.e., if a human talks about a 'Car', but further mentions specific attributes (e.g., the color 'Red'), it is clear to his communication partner, which type of car is meant (i.e., the 'E3'). This process should be mimiced by the concept.

3) Humans sometimes only mention specific attributes of what they actually refer to, i.e., in contrast to the previous requirement they don't mention the 'Car' concept, but may only refer to the car by one of its attribute. An example could be 'I drive a red one'. Still the listener knows what the speaker most probably meant (a car or here again the 'E3'). The concept should try to identify and solve this problem.

The last two requirements can be summed up by saying that although some concepts might not be linked to the correct word or the semantic relation is missing between two concepts it should still be possible to identify the actually meant concepts of the user. Such a task is difficult to achieve. Usually algorithms 'only' identify the most likely concepts for a given text out of a set of directly available concepts. Since many algorithms are based on WordNet only, domain specific information might not be available, which could indicate a relation between 'Car', 'E3' and 'Red'. Statistical WSD concepts, which rely on n-grams might in some cases be capable of handling this problem. However they require that a fact has to be stated at least once in textual form to correctly disambiguate a specific context.

V. Concept

The algorithm is separated into three different phases: Initialize tokens, create token flow and analyse token flow. All phases will be explained in the following sections.

A. Definitions

For our concept we need an Ontology $O := (C, R, G)$, where C is a set of concepts, R defines a set of relations between the concepts in C and G defines a set of generalizations links between the concepts in C. The algorithm is initialized using an **input** $I := (c_s, c_y, c_t, S_c)$, where c_s is the source concept, c_y is the concept of a relation, which has c_s as its source (e.g., 'Drive' would be the concept of a relation between 'Driver' and 'Vehicle') and c_t specifies the target of the relation of c_y. Finally $S_c := c_1..c_n$ is a set of further concepts, which act as additional information (context) to the spreading process. I can also consist of (c_s, c_y) or (c_s, c_t) only. S_c is always optional.

A **token container** a is defined by the tuple (c, T, act, d). a is associated with a concept $c \in O$ (this is also the ID of the token container) and a set of tokens $T := t_1..t_n$. It basically acts as a container for all the tokens, which have reached the specific concept c. It further contains an attribute act (we will refer to attributes like $a.act$ in the following), which indicates if the concept $a.c$ has been a part of the spreading activation input I (if we talk about a being part of I or another set of concepts in further references, we actually mean $a.c$, which should be contained in the corresponding concept set). d represents the depth of the tokens concept c within the ontologies generalization hierarchy. The depth value is calculated as the position of c relative to the length of the longest branch it is located in. In the following we will refer to a_s as the container of c_s, a_t as the container of c_t and a_y as the container of c_y.

A **token** t is defined by the tuple $(orig, start, pos, e, s, dir)$. $t.orig$ holds a reference to its original container (this must be a container of one of the concepts in I). Next, it contains a reference $t.start$ to the container where it originally started from (this can, but does not have to be the original container; it may also be a container whose concept is related to the concept of $t.orig$ via generalization). $t.pos$ is the container, which represents the current position of the token. $t.e$ indicates the remaining energy of the token (if the energy drops below a certain threshold this token can not spread any further). $t.s$ describes the steps the token has already traveled within the ontology. $t.dir$ defines the direction a token is traveling in. Values can be up / down (within the generalization hierarchy) or sidewards (i.e., on an association).

B. Initialize tokens

The algorithm is initialized based on each $c \in I$ with Algorithm 1 (e.g., $INIT(c_s, 1.5)$). As can be seen the initialization is based on the generalization hierarchy of the corresponding concept. All concepts of I are basically treated the same (i.e., their energy value is the same). The only exception is c_s, which receives a higher initial energy value than the remaining elements. The cause for this is that we especially want to know if there is a path from the source to the target concept. Therefore tokens from c_s receive a higher energy, which allows them to travel further.

As can be seen in algorithm 1 the initialization is done going in both generalization directions ($INITGENUP$ means that the initialization is done up the generalization hierarchy, i.e., more general elements are initialized, whereas $INITGENDOWN$ initializes more specific elements). This is done because humans tend to be ambiguous while communicating and often use more generalized terms than they actually mean (see requirement 2 in IV). Only the context of a word helps in deciding, which concept they actually refer to. Therefore, the call down the hierarchy helps to initialize all elements, which eventually are meant by a human. In contrast the call upwards initializes all those elements, which may contain the corresponding semantic information that the current concept c inherited from them. This information is necessary in order to correctly analyze the current input.

$INITGENUP$ initializes a single concept and its generalization hierarchy upwards by creating a container for every concept in the upwards generalization hierarchy and further creating the initial tokens for each of these concepts ($INITGENDOWN$ works analogously). It is important that every concept, which will be reached by a call of $INITGENUP$ in the generalization hierarchy is treated as being a part of the original input. Therefore the $a.act$

attribute of their containers will be set to true. The cause for this is that each of these concepts could be the carrier of the information we will later on be searching for.

Algorithm 1 Initialization

```
procedure INIT($c$, $ENERGY$)
    INITGENUP($c$, $c$, $ENERGY$, $null$)
    INITGENDOWN($c$, $c$, $ENERGY$, $null$)
end procedure
```

C. *Create token flow*

The set of initial tokens has been created. Now the token flow itself has to be generated. The overall process is shown in algorithm 2. As can be seen the process itself is discretized in single phases. Each current token generation $T_{current}$ leads to a new token generation T_{next}, which will only be processed after every token from the current generation has been processed. This methodology is important as the $POSTPROCESS$ call initializes a back propagation mechanism. A non discretized process would yield indeterministic results.

$CREATETOKENFLOW$ gets the set of current as well as next tokens. For every single token in $T_{current}$ it does the following: First it checks if the $t.pos$, $t.dir$ and $t.e$ attributes allow a next step. If $t.dir$ is unknown, it is allowed to travel both on associations (sidewards) as well as on generalizations (up / down). A token is however not allowed to go up, if it was going down before. Also it may not go up if it was going sidewards before. The cause for these restrictions is that the tokens elseway could reach not necessary or false concepts.

Next new tokens are being generated for the next step of the current token (i.e., tokens for the relation itself as well as the target of the relation) and added to the T_{next} set. The energy of the new tokens is based on the current tokens $t.e$ attribute and is being decreased by a fixed value. However if the container of the relation has been activated (i.e., $a.act == true$), no energy will be subtracted from the energy of the new token. This process allows us to enhance the energy of paths, which are likely to be more relevant to the spreading activation input.

Next the $POSTPROCESS$ method is called. It starts the back propagation mechanism on all containers whose $a.act$ attribute is set to true and have received new tokens in the last token flow phase. Each token on such a container then gains an increase of its energy value: $t.e = t.e + (E_{MAX} - t.e) * C_e$, where E_{MAX} denotes the maximum energy a token can have and C_e is a constant factor between 0 and 1. This mechanism is recursively continued on the predecessor of this token. By activating the propagation mechanism on such containers, which are probably relevant to the input (again $a.act == true$), only such token path are strengthened, which seem to indicate the most likely results. The cause for this is that the concepts we search for are most likely closely connected (i.e., there are only few relations and therefore few steps to get from one concept to another) and also super- or subtypes of the original input concepts c_s, c_y and c_t.

Algorithm 2 Process Tokens

```
procedure PROCESSTOKENS
    while $T_{next}.size \neq 0$ do
        $T_{current} \leftarrow T_{current} \cup T_{next}$
        $T_{next} \leftarrow \{\}$
        PREPROCESS
        CREATETOKENFLOW($T_{current}, T_{next}$)
        POSTPROCESS
        $T_{current} \leftarrow \{\}$
    end while
end procedure
```

D. *Analyze token flow*

The final step consists of gathering the results from the token flow process. We first start by identifying more specific elements of the actual input (see Section IV). For this we first collect all containers for every $c \in I$, which are more specific than c. Next we sort them based on the number of relevant tokens, which arrived there (i.e., tokens from concepts of I/c), their token weight (higher is better), activation times (i.e., how often the container was activated in the $POSTPROCESS$ method, more is better) and the depth of their concept (deeper is better). We then pick the best element from this list. This then is the more specific element of c_m. However in case that we find too many elements, which might be relevant to our criteria we don't pick any elements as this would contradict the idea of specifying the initial input.

All information necessary for the final result has been computed. However it might be the case that this result might not be perfect, i.e., the initial input was ambiguous (because of ambiguous statements of a human speaker, e.g., requirement three in Section IV). For such a situation we have developed a heuristic, which identifies this case and tries to identify a better solution. First there are however some restrictions to be made: Such an 'imperfect' situation can only be identified if c_s, c_y and c_t are provided in I. In other cases there would be too few information, which would lead the heuristic to imprecise decisions. Further only situations in which either c_s or c_t are wrong can be detected. For the following we will use the example from Section IV in, which case c_t is wrong (as it references 'Red' instead of 'Car').

We first collect all available associations, which are of type c_y and reference c_t. Those are stored in a list A_p. A_p is then sorted based on the weight of the associations source and target container weights (i.e., the weight of the containers based on the tokens, which arrived there). Now the algorithm looks if one of the associations in A_p has a source and a target, which matches c_s or c_t: $(r.s \subseteq c_s \vee r.s \supseteq c_s) \wedge (r.t \subseteq c_t \vee r.t \supseteq c_t)$, where $r.s$ is the source concept of a relation of A_p and $r.t$ is the target concept. If this is the case the algorithm seemingly has been used on a correct input and the spreading activation is finished. If however no association of A_p matches this condition, the algorithm will be reinitialized. For this the best association of A_p (i.e., the one with the highest source and target container weights) is used because based on the current token flow this association has been marked as the best possible match. Now the spreading activation is reinitialized with a new I':

1) The 'wrong' concept (either c_s or c_t) will be replaced with the new concept ($r.s$ or $r.t$) of the best association of A_p (in our example this means that c_t 'Red' will be replaced with $r.t$ 'Car'. A more elaborate example will be given in Section VI).
2) The old element (c_s or c_t) will be added to the list of context elements, as it might provide helpful information for the next spreading activation iteration. This is done because the user might have had a reason to mention this specific concept initially therefore the concept is not thrown away, but used as a context concept).

Regarding the example from Section IV, I was $(Person, Drive, Red, \{\})$ and I' is now $(Person, Drive, Car, \{Red\})$. With I' the process is now being restarted and the same steps are applied as described before. If in this second iteration a seemingly correct result could be found the algorithm will return it. If however the conditions for starting the heuristic would match again, we stop the process. We then return the best result from both iterations. This has proved to provide good results.

Finally a value is computed, which indicates if the information we searched for exists within the ontology. There are two different cases to be distinguished:

1) The first case occurs, if the heuristic did not step in, i.e., the initial source and target elements are still the same. Then a token t from a_t is searched, which has $t.start == a.s$, i.e., it happens to have the source container as its starting position. If such a token could be found the computation of the final value depends on the average energy of the token regarding the length of the token path (excluding the generalization).
2) The second case happens if the original source or target containers have been exchanged for a new container. If this is the case, the value depends on the semantic similarity (based on a lowest common ancestor approach) between the initial c_s / c_t concepts from I and the current, 'new' c'_s / c'_t from I' concepts.

VI. Case Study

Due to the structure of our concept there are no known gold standards for our case, as existing ones like Senseval or Semeval are difficult to use for us. Senseval-2 for example provides texts, which have been annotated with WordNet 2. However WordNet is a lexical database and therefore mainly contains linguistic information, not domain relevant semantic information. Other stochastically motivated test data is not suited for our scenario at all. We can therefore not provide any elaborate statistical evaluations yet. Therefore we focus on some actual examples from our test scenario.

Our scenario currently consists of an ontology with about 100 concepts. We will show some different examples in detail in the following section. Figure 1 shows a simplified excerpt from this ontology. Its structure describes a simple car domain, which contains drivers (driving cars), different cars with different colors (E2, E3), another car E1, which has problems with its engine. Further a CEO is supposed to drive specific cars (the E2 and E3).

The first request will show the resolution of overgeneralization. 'Driver Drives E2' is supposed to detect if the concept 'Driver' is related to 'E2' using a relation of type 'Drives'. As can be seen in the picture there is a 'Drives'-relation from 'Driver' to 'Car', which is the supertype of E2. However there is also a more specific information, which could state exactly the same and in this case is even shorter: 'CEO Drives E2'. As the 'CEO' is a subconcept of 'Driver', it will be activated in the inialization phase and will itself spread tokens. As 'Drives' is also activated the token will pass with no loss of energy to 'E2'. The same is the case for the token, which will arrive at 'E2' from 'Driver'. However, this one needed more steps for its 'journey'. After the spreading activation has finished the algorithm checks every initial starting element for more concrete information. 'Driver' is the only concept, which contains a subconcept. As there are enough hints (due to backpropagation as described in Section V) that 'CEO' might be a better suited alternative to the initial request, the algorithm proposes 'CEO' as an alternative for 'Driver' to the user. As there is a direct relation available, the semantic value of the request is calculated to be 1.

A more complex request is the triple 'Driver Drives Red', i.e., a concept 'Driver' is connected to a concept 'red' using a relation of the type 'Drives' (such a request could be the case in a sentence like 'The driver drives a red one'). As can be seen in Figure 1, 'Car' is related to color and 'E3' is related to 'Red'. If the spreading activation starts the tokens will spread through the network and due to backpropagation the 'Car' concept receives a significantly higher energy than the remaining elements, as it is part of an important path between 'Driver' and 'Red' / 'Color'. As we are searching for a triple the algorithm 'sees' that there is no direct relation available between 'Driver' and 'Red'. However the 'Car' element could be matching, as one relation has 'Driver' as its start concept and 'Drive' as its type. Therefore the algorithm reinitializes itself and replaces the 'Red' element with the 'Car' element ('Red' becomes a context element). In the second pass tokens from 'Driver' as well as 'CEO' will reach 'Car' as well as 'E3'. Tokens from 'E3' will reach 'Red'. Backpropagation will then again lead to an increase of energy in 'E3' and 'CEO'. As the algorithm could successfully solve the initial request it proposes 'E3' instead of 'Red' and 'CEO' instead of 'Driver'. The semantic similarity of the request however is weighted with 0.75 because we can not be absolutely sure that the user really meant 'Car' with 'Red'.

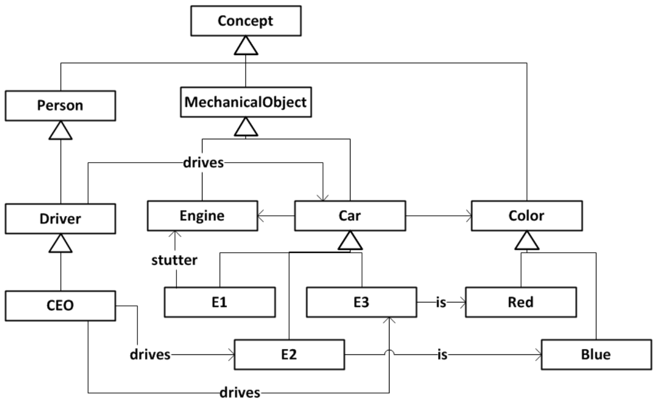

Figure 1. Example of our ontology

VII. Conclusion

The biggest problem is the knowledge acquisition problem as it is the case with every knowledge intensive system. Especially the creation of an ontology, which provides a good representation of the corresponding domain is a huge problem. We try to tackle this one by creating a corresponding set of tools and workflows, which allow an easy and semi-automatic process for this task.

In this paper we have presented a spreading activation based algorithm, which works directly on a domain ontology without creating its own SAN. It helps us in solving the WSD problem and in certain cases also proposes concepts, which are more likely to be meant instead of the initial input concepts.

The algorithm is still ongoing work and its prototypical implementation is constantly being used within our framework for creating semantic interpretations of natural language text. As such it delivers good results in our scenarios. Especially its feature of 'guessing' better suited concepts greatly helps in interpreting natural language text with all its ambiguities.

References

[1] P. Hitzler, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, "OWL 2 Web Ontology Language Primer," *Director*, no. October, pp. 1–123, 2009. [Online]. Available: http://www.w3.org/TR/2009/REC-owl2-primer-20091027/

[2] W. Ontology, "OWL 2 Web Ontology Language Document Overview," *October*, vol. 2, no. October, pp. 1–12, 2009. [Online]. Available: http://www.w3.org/TR/owl2-overview/

[3] W. Fischer and B. Bauer, "Combining Ontologies And Natural Language," *Proceedings of the Sixth Australasian Ontology Workshop*, 2010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21409794

[4] W. Fischer and B. Bernhard, "Cognitive-Linguistics-based Request Answer System," in *Adaptive Multimedia Retrieval. Understanding Media and Adapting to the User*. Madrid: Springer, 2011, pp. 135–146. [Online]. Available: http://www.springerlink.com/content/l426675knx75765m/

[5] J. Kleb and A. Abecker, "Entity Reference Resolution via Spreading Activation on RDF-Graphs," *The Semantic Web Research and Applications*, vol. 6088, pp. 152–166, 2010. [Online]. Available: http://www.springerlink.com/index/10.1007/978-3-642-13486-9

[6] G. Tsatsaronis, M. Vazirgiannis, and I. Androutsopoulos, "Word Sense Disambiguation with Spreading Activation Networks Generated from Thesauri," in *IJCAI 2007*, M. M. Veloso, Ed., 2007, pp. 1725–1730. [Online]. Available: http://dblp.uni-trier.de/rec/bibtex/conf/ijcai/TsatsaronisVA07

[7] G. Tsatsaronis, I. Varlamis, and M. Vazirgiannis, "Word Sense Disambiguation with Semantic Networks," *Work*, vol. 5246, pp. 219–226, 2008. [Online]. Available: http://www.springerlink.com/content/87p101317131078t

[8] G. Tsatsaronis, I. Varlamis, and K. Nø rvåg, "An experimental study on unsupervised graph-based word sense disambiguation," *Computational Linguistics and Intelligent Text Processing*, pp. 184–198, 2010. [Online]. Available: http://www.springerlink.com/index/N577Q110122R04J6.pdf

[9] E. Agirre, A. Soroa, and M. Stevenson, "Graph-based word sense disambiguation of biomedical documents." *Bioinformatics*, vol. 26, no. 22, pp. 2889–2896, 2010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/20934991

[10] S. Kang and J. Lee, "Ontology-based word sense disambiguation using semi-automatically constructed ontology," *MT Summit VIII Machine Translation in the Information Age Proceedings Santiago de Compostela Spain 1822 September 2001 pp181186 PDF 287KB*, 2001.

[11] T. Hussein, D. Westheide, and J. Ziegler, "Context-adaptation based on Ontologies and Spreading Activation," *Citeseer*, 2005.