Contextual Vocabulary Acquisition:

CASSIE Learns to Figure Out a Meaning of

an Adjective *Importunate* from Context

Nyurguyana Petrova

CSE727 Contextual Vocabulary Acquisition

petrova3@buffalo.edu

May 10, 2007

Abstract

Contextual Vocabulary Acquisition (CVA) is the deliberate acquisition of a meaning of an unknown lexical item from a given text with the help of prior knowledge of a reader and context without consulting dictionaries and people. This paper is a report on the contextual vocabulary acquisition project - representation of the adjective *importunate* in SNePS. The representation of an unknown word was based on information attained from a human subjects. Henceforth, a meaning of a word importunate is purely based on the beliefs and prior knowledge of subjects who participated in the experiment. As a result of the project SNePS is able to infer a meaning of an unfamiliar word - *importunate*.

1 Contextual Vocabulary Acquisition Project and SNePS

A natural language is a rich and dynamic process which among other diachronic as well as synchronic changes entails a constant change of vocabulary. Due to these constant changes it is almost impossible for a human being to know all the words existing in a language. Therefore, when reading a book or a newspaper it is inevitable for a person to run into a word that he/she has never seen before. The solution to this problem is either to consult a dictionary for a meaning of an unknown word or to continue on reading hoping that an unknown word has a minor effect on understanding a passage that one is reading. Most of the time this is what people usually do. But what if the understanding of an unfamiliar word is essential in comprehending a text? The strategy suggested by most of the researchers in areas such as psychology and first-language acquisition is to simply *guess*. Rapaport & Kibby (2002) argue that this methodology is rather vague and lacks a detailed explanation of the overall procedure, which is the result of not knowing how exactly context operates [Rapaport & Kibby, 2002]. In order to able to use a context successfully one needs to know how context operates and how it can be used in figuring out a meaning of an unknown word. The Contextual Vocabulary Acquisition (henceforth, CVA) project is an ongoing research project conducted by William J. Rapaport and Michael W. Kibby at the University at Buffalo (SUNY). The CVA project is concerned with the problem of the deliberate acquisition of a meaning of an unknown lexical item from a text with the help of prior knowledge of a reader and context without consulting any dictionaries or people [Rapaport, 2003]. The main goal of their research is "to develop a computational theory of CVA [as well as] to adapt the strategies for doing CVA (embodied in [their] algorithms) to an educational curriculum for teaching CVA strategies to students in classroom settings" [Rapaport, 2003, p 1]. The CVA project uses SNePS in achieving the above mentioned goals. SNePS is a network-based knowledge representation, reasoning and acting system [5]. According to Shapiro and Rapaport (1987),

> SNePS is a semantic network language with facilities for building semantic networks to represent virtually any kind of information, retrieving information from them, and performing inference with them [Shapiro & Rapaport, 1987, p 263].

Users of SNePS are able to interact with it via different kinds of languages such as a Lisp-like user language and a higher-order logic language [Shapiro & Rapaport, 1987, p 263]. The information in SNePS is represented by nodes. The underlying syntactic structure is based on the arcs, which come out as directed arcs from nodes and represent relations between the nodes [Shapiro & Rapaport, 1987, p 266]. In the CVA project, the users interact with the cognitive agent CASSIE (the cognitive agent of the SNePS system-an Intelligent Entity). The process of teaching CASSIE to acquire new words amounts to the similar process of teaching human beings to learn to figure out a meaning of an unfamiliar word from a given context and a knowledge base. The strategic development of the overall procedure of acquiring an unknown word with the help of the cognitive agent will allow us to come up with a better methodology to teach human beings to learn to acquire a new vocabulary.

2 The Passage

The purpose of this project is to build a knowledge representation based on a short passage, which is Jane Smiley's introduction to the novel *Of Human Bondage* by Somerset Maugham. With the help of the developed knowledge base, CASSIE should be able to figure out a meaning of the word *importunate* from context. The task gets more challenging due to the fact that a word *importunate* is an adjective as a part of speech. Although the base algorithms for nouns and verbs are implemented and are in the process of improvement, the implementation of an adjective is still a challenge for the CVA project. According to Rapaport & Kibby (2002), "adjectives are much less susceptible to CVA than nouns or verbs" [Rapaport & Kibby, 2002]. Since the ultimate goal for the CVA project is to improve vocabulary acquisition by creating a curriculum to use CVA, this project should be considered as one of the minor steps in establishing heuristics for figuring out the meanings for adjectives. The target word and its context that we are discussing in this paper are given below:

> *For one thing, Philip's affair with Mildred is not his first sexual experience. His sojourn at the university in Heidelberg, Germany, at the age of eighteen, has exposed him to a world somewhat less sexually repressed than the English society where he has spent his adolescence, and this new*

knowledge, in turn, has rendered him vulnerable to the advances of a woman friend of his uncle and aunt [mwk–the uncle is a minister in a rural, conservative area of England], Miss Wilkinson, herself experienced in the ways of the Continent. Though thirty-seven, Miss Wilkinson looks younger, and at first fascinates and attracts Philip, who would like to think she can't be more than twenty-six, only six or seven years older than he is. As she becomes more **importunate**, *though, Philip begins to feel discomfort and physical revulsion in her presence, and he is only too glad when she departs for governess position in Berlin.*[1]

Because the task of understanding a meaning of an adjective is rather difficult and more challenging than those of nouns and verbs due to its quality description nature, the necessary amount of previous context is essential. Although most of the information provided in the passage is important for understanding a meaning of the word *importunate*, the information about time and aspect was not captured when representing the knowledge base. The sentence where the target word *importunate* occurs was elaborated and analyzed into three simpler parts:

As she becomes more importunate, though, Philip begins to feel discomfort and physical revulsion in her presence, < ... >.

1). When Philip is in Miss Wilkinson's presence, he begins to feel discomfort and physical revulsion.

2). Miss Wilkinson becomes (is) importunate.

3). Philip feels discomfort and physical revulsion as Miss Wilkinson becomes (is) more importunate.

Miss Wilkinson's becoming importunate is represented as a static property of being importunate since it does not make much difference for the understanding of the word *importunate*. The fact that Philip feels discomfort and physical revulsion was regarded as the effect of Miss Wilkinson having the property importunate.

[1] Kibby, Michael (2006), CVA Texts for Rapaport's seminar, http://www.cse.buffalo.edu/~rapaport/CVA/CVATextsRapaportSeminar.pdf

3 "Think-Aloud" Protocols

The original passage with the target word replaced by an artificial word *onrustamous*[2] was given to seven human subjects for reading: four of them were native English speakers, and three of them were non-native English speakers. The human subjects were graduate students and professors from the Department of Linguistics at the University at Buffalo. Their age ranged from 23 to 45. The procedure of the experiment was as following:

1. A human subject was asked to read the whole passage from a piece of paper.

2. A human subject was asked to find a word that he/she was not familiar with and did not know a meaning of.

3. The next step for a human subject was to try to define the unfamiliar word. In the process of defining an unknown word, the subject was asked to "think-aloud", so that the reasoning that led to a hypothesis was explicit.

4. A person was asked to point out a sentence or a set of sentences that helped him/her to hypothesize a meaning for a word.

Below are the examples showing some answers and comments of the protocols (see Appendix 2):

- Subject 1: "... more open with ones feelings towards someone; more pushy. This sentence shows that she was sort of pushy '< ... > the English society has rendered him vulnerable to the advances of a woman < ... >'. The following sentence shows that she was older than Philip 'Though thirty-seven, Miss Wilkinson looks younger, and at first fascinates and attracts Philip, who would like to think she can't be more than twenty-six, < ... >'. And Philip is younger than Miss Wilkinson '< ... > Philip, who would like to think she cant be more than twenty-six, only six or seven years older than he is.' And also the sentence '< ... > she departs for governess position in Berlin.', he is leaving that is why she needs to develop the relationship quickly."

[2]The artificial word *onrustamous* was suggested by William Rapaport

- Subject 2: "...sexually aggressive. Because it says '< ... > the English society < ... > has rendered him vulnerable to the advances of a woman < ... >', and also '< ... > Philip begins to feel discomfort and physical revulsion in her presence < ... >'."

As a result of the experiment the target word *importunate* was defined as:

- more open with one's feelings towards someone, more pushy;
- sexually aggressive;
- interested in someone;
- acting older or seeming more mature, than prior, to an outside observer;
- opposite of hard to get, lustful/nymphomaniac;
- pressing, advancing on someone;
- familiar, visible.

When defining an unfamiliar word, the human subjects seem to point out mostly two facets of a possible definition: *more pushy*, on the one hand, and *sexually aggressive*, on the other. According to Roget's New Millennium Thesaurus[3] a word *importunate* is defined as *demanding*; the Random House Unabridged Dictionary[4] defines it as *urgent or persistent in solicitation, sometimes annoyingly so.* The definition *more pushy* assumed by the human subjects is closer to the standard definition of the word *importunate* given by dictionaries. However, most of the human subjects seem to agree that the unfamiliar word may be interpreted as *sexually aggressive.* The human subjects were also consistent with the sentences that helped them to figure out a possible meaning for the unknown word (see Appendix 2). When developing a knowledge base for CASSIE, we are going to assume that the word *importunate* is interpreted as *more pushy* and *sexually aggressive.*

[3] http://thesaurus.reference.com/browse/importunate

[4] http://dictionary.reference.com/browse/importunate

4 Syntax and Semantics of the SNePS Case Frames

In this project, eight case frames were employed: seven of them are typical case frames, the syntax and semantics of which are given on the CVA home webpage.[5] The case-frame which is not typical and which is not described in the CVA webpage yet is a *cause/effect* case frame. The list below shows the case frames that were used in this particular project:

- member/class
- object/property
- object/propername
- object/location
- lex
- object1/rel/object2
- cause/effect
- agent/act/action/object

One of the examples of the *cause/effect* case frame is represented in Figure 1. The semantics of this case frame is given below:

```
Cause/effect = [m11!] is the proposition that effect [p18] is caused by
[p17].
```

5 Background Knowledge

The background knowledge is an indispensable component for CASSIE in figuring out a meaning of an unknown word from the passage. That is why the formalization of the background knowledge should be well thought out to avoid any possible ambiguities. Below is the knowledge base given to CASSIE prior to

[5] http://www.cse.buffalo.edu/~rapaport/CVA/cva.html

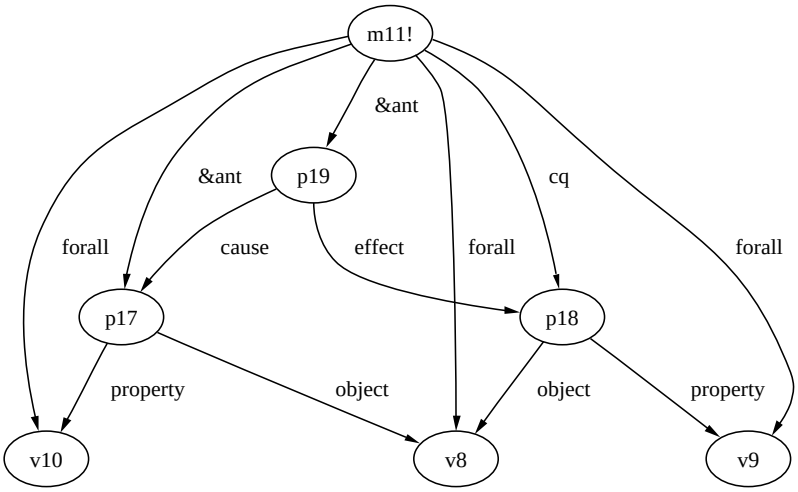

Figure 1: CAUSE/EFFECT

showing the sentences from the passage. The background knowledge is represented by the several asserted rules annotated for ease of reading and understanding.

```
;;;=============================================================================
;;;If someone is a person, and that person is in discomfort and in physical revulsion,
;;;then he/she is not ok.
;;;=============================================================================
(describe (assert forall $x
&ant ((build member *x class (build lex "person"))
      (build object *x property (build lex "discomfort"))
      (build object *x property (build lex "physical revulsion")))
cq (build object *x property (build lex "not-ok"))))
;;;=============================================================================
```

The rule below can be elaborated as follows: If *importunate* is an unknown concept & if a person becomes importunate, & if a second person is not OK because the first person became importunate, then the first

person is pushy because that person is importunate.

```
;;;=============================================================================
;;;If X is an unknown concept & if a person becomes X
;;;& if a second person is not OK because the first person became X,
;;;then the first person is pushy because that person is X.
;;;=============================================================================
(describe (assert forall ($x $p $q)
&ant ((build object *x property (build lex "unknown"))
              (build object *p property *x)
              (build object *q property (build lex "not-ok"))
              (build cause (build object *p property *x)
     effect (build object *q property (build lex "not-ok"))))
cq (build cause (build object *p property *x)
  effect (build object *p property (build lex "pushy")))))
;;;=============================================================================
```

The following rule could be interpreted as follows: If importunate is an unknown concept, & if a person becomes importunate, & if a second person is not OK because the first person became importunate, then the first person is sexually aggressive because that person is importunate:

```
;;;=============================================================================
;;;If X is an unknown concept
;;;& if a person becomes X
;;;& if a second person is not OK because the first person became X,
;;;then the first person is sexually aggressive because that person is X.
;;;=============================================================================
(describe (assert forall ($x $p $q)
&ant ((build object *x property (build lex "unknown"))
```

```
               (build object *p property *x)
               (build object *q property (build lex "not-ok"))
               (build cause (build object *p property *x)
     effect (build object *q property (build lex "not-ok"))))
cq (build cause (build object *p property *x)
     effect (build object *p property (build lex "sexually aggressive")))))
;;;=============================================================================
```

The following recursive rule was intended to formalize the prior knowledge where it states if a person is pushy (or sexually aggressive) because a person is importunate, & if a person is importunate, then a person is pushy (or sexually aggressive):

```
;;;=============================================================================
;;;If p is f because p is x
;;;& if p is x
;;;then p is f
;;;=============================================================================
(describe (assert forall ($p $f $x)
  &ant ((build cause (build object *p property *x)
     effect (build object *p property *f))
               (build object *p property *x))
  cq (build object *p property *f)))
;;;=============================================================================
```

The knowledge of *someone making advances towards another person while being and older than that other person, which implies that a former is pushy and sexually aggressive towards a latter one* is represented in the following rule:

```
;;;=============================================================================
;;;Forall x y [x makes advances towards y
```

```
;;;& x is older than y
;;;then x is pushy and sexually aggressive towards y]
;;;=============================================================================
(describe (assert forall ($x $y)
&ant(build agent *x act (build action (build lex "makes advances towards") object *y))
&ant(build object1 *x rel (build lex "is older than") object2 *y)
cq(build object1 ((build object *x property (build lex "pushy"))
  (build object *x property (build lex "sexually aggressive")))
 rel (build lex "towards")
 object2 *y)))
;;;=============================================================================
;;;Forall x y [x is pushy & sexually aggressive towards y
;;;then y feels discomfort and physical revulsion
;;;because x is pushy and sexually aggressive towards y]
;;;=============================================================================
 (describe (assert forall ($x $y)
&ant((build object *x property (build lex "pushy"))
     (build object *x property (build lex "sexually aggressive"))
     (build object1 ((build object *x property (build lex "pushy"))
           (build object *x property (build lex "sexually aggressive")))
    rel (build lex "towards")
    object2 *y))
 cq (build cause (build object1 ((build object *x property (build lex "pushy"))
                                 (build object *x property (build lex "sexually aggressive")))
                          rel (build lex "towards")
                          object2 *y)
           effect ((build object *y property (build lex "discomfort"))
```

```
                    (build object *y property (build lex "physical revulsion"))))))

;;;============================================================================
```

The following rule is formalized based on the assumption from the passage that y is Philip, x is Miss Wilkinson, and f is an unknown concept which in this case is importunate.

```
;;;============================================================================
;;;Forall y x f [y feels discomfort and physical revulsion
;;;because x is pushy and sexually aggressive towards y
;;;& y feels discomfort and physical revulsion because x is f
;;;& f is unknown
;;;then
;;;Forall p1 [p1 is f
;;;then p1 is pushy and sexually aggressive]]
;;;============================================================================
 (describe (assert forall ($y $x $f)
&ant (build cause (build object1 ((build object *x property (build lex "pushy"))
 (build object *x property (build lex "sexually aggressive")))
        rel (build lex "towards")
        object2 *y)
    effect ((build object *y property (build lex "discomfort"))
    (build object *y property (build lex "physical revulsion"))))
&ant(build cause (build object *x property *f)
   effect ((build object *y property (build lex "discomfort"))
   (build object *y property (build lex "physical revulsion"))))
&ant(build object *f property (build lex "unknown"))
cq(build forall $p1
   ant(build object *p1 property *f)
```

```
   cq((build object *p1 property (build lex "pushy"))
      (build object *p1 property (build lex "sexually aggressive"))))))
;;;============================================================================
```

6 A SNePS Representation of Sentences from the Passage

The information provided in the passage that was essential for understanding the unfamiliar word *importunate* was also presented to CASSIE. Most of the important information was taken from the sentences that were pointed out by the verbal protocols. The sentences were simplified but they are still able to deliver the gist of what the author wanted to say.

```
;;;============================================================================
;;;Someone whose name is Philip
;;;============================================================================
(describe (add object #philip propername "Philip"))
;;;============================================================================
;;;Someone whose name is Wilkinson
;;;============================================================================
(describe (add object #wilkinson propername "Wilkinson"))
;;;============================================================================
;;;Philip and Miss Wilkinson are persons.
;;;============================================================================
(describe (add member (*philip *wilkinson) class (build lex "person")))
;;;============================================================================
;;;Miss Wilkinson makes advances towards Philip.
;;;============================================================================
(describe (add agent *wilkinson act
(build action (build lex "makes advances towards") object *philip)))
```

```
;;;=============================================================================

;;;Miss Wilkinson is older than Philip.

;;;=============================================================================

(describe (add object1 *wilkinson rel (build lex "is older than") object2 *philip))


;;;=============================================================================

;;;SNePS representation of the sentence from the text:

;;;As she becomes more importunate, though,

;;;Philip begins to feel discomfort and physical revulsion in her presence, <.>.

;;;

;;;1). When Philip is in Miss Wilkinson's presence,

;;;    he begins to feel discomfort and physical revulsion.

;;;2). Miss Wilkinson becomes (is) importunate.

;;;3). Philip feels discomfort and physical revulsion

;;;    as Miss Wilkinson becomes (is) more importunate.


;;;From Maugham W. Somerset. (1915, 1991). {\em Of Human Bondage}.

;;;=============================================================================

;;;When Philip is in Miss Wilkinson's presence,

;;;he begins to feel discomfort and physical revulsion.

;;;=============================================================================

(describe (add

        &ant ((build object *philip location #someplace)

              (build object *wilkinson location *someplace))

        cq ((build object *philip property (build lex "discomfort"))

            (build object *philip property (build lex "physical revulsion")))))
```

```
;;;=============================================================================

;;;Philip and Miss Wilkinson are in the same place.

;;;=============================================================================

(describe (add object (*philip *wilkinson) location *someplace))

;;;=============================================================================

;;;Miss Wilkinson becomes (is) importunate.

;;;=============================================================================

(describe (add object *wilkinson property (build lex "importunate")))

;;;=============================================================================

;;;As Miss Wilkinson becomes importunate, Philip feels discomfort and physical revulsion.

;;;=============================================================================

(describe (add cause (build object *wilkinson property (build lex "importunate"))

       effect ((build object *philip property (build lex "discomfort"))

       (build object *philip property (build lex "physical revulsion")))))

;;;=============================================================================

;;;Philip is not ok because Miss Wilkinson is importunate.

;;;=============================================================================

(describe (add cause (build object *wilkinson property (build lex "importunate"))

               effect (build object *philip property (build lex "not-ok"))))

;;;=============================================================================

;;;Word (importunate) is unknown.

;;;=============================================================================

(describe (add object (build lex "importunate") property (build lex "unknown")))

;;;=============================================================================
```

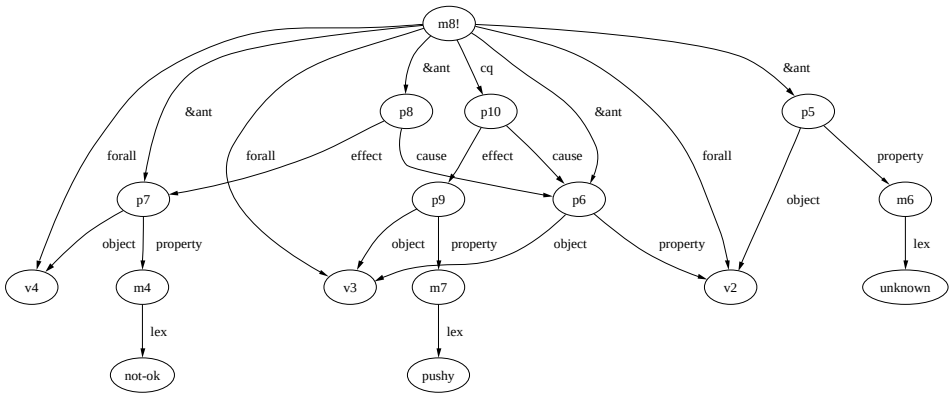

Figure 2: PROPERTY: Pushy

7 Results

With the help of the background knowledge and a SNePS representation of sentences from the passage CASSIE is able to produce a network (see Figure 5), which is a representation of the unknown word *importunate* in Cassie's mind. Because the algorithm for adjectives is not developed yet, it is impossible to query CASSIE to define an unfamiliar word. Instead with the help of the *show* command one can easily see what kind of representation of a word CASSIE has built up in her mind. As can be seen in Figure 5, where a broader representation is presented, CASSIE was able to connect the properties *pushy* and *sexually aggressive* with the unknown word *importunate*. In other words, with the help of the prior knowledge, rules and the sentences from the text she was able to define a previously unknown word *importunate*. Generally speaking, the rule presented as asserted node m8! in Figure 2 states *'A person who has a property unknown has also property pushy'*. In Figure 3 the rule represented with m10! says *'A person who has a property unknown has also property sexually aggressive*. And in Figure 4 m59! states that *'a word* importunate *is unknown concept'*. Figure 5 shows the nodes (namely m59!, m60!, m61!, m62! and m63!), which is a result of CASSIE building an inference to the last input 'Word is unknown'. It states *'an unknown word is importunate; and if someone is importunate then he/she is pushy and sexualy aggressive'*

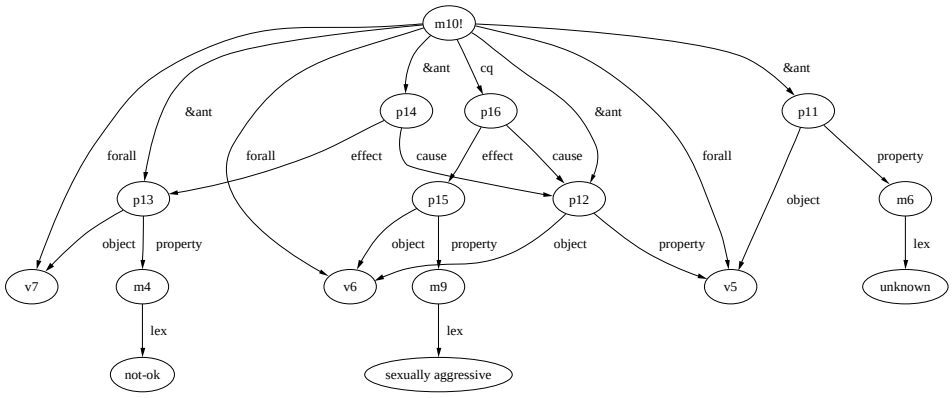

Figure 3: PROPERTY: Sexually Aggressive

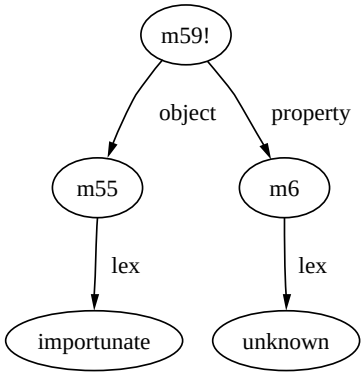

Figure 4: A word importunate is unknown concept.

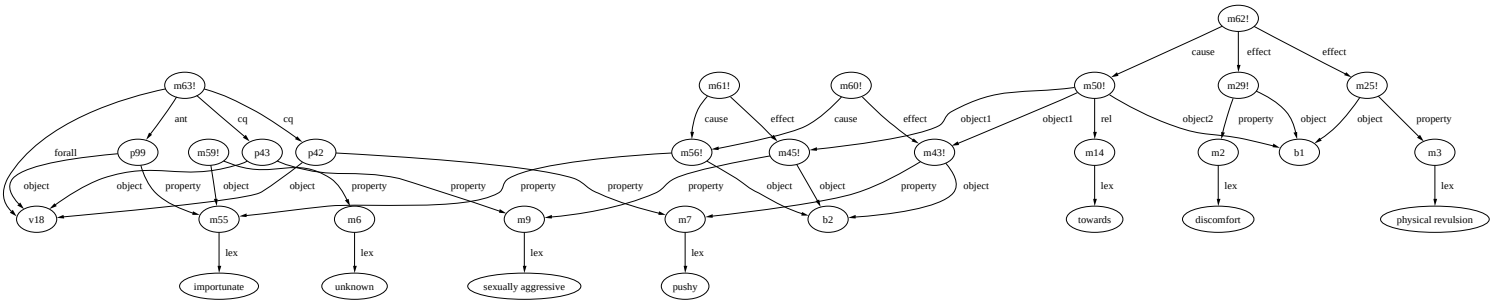

Figure 5: Representation of a word *importunate*

8 Future Study

A word importunate is an interesting lexical item to work with. In addition to falling under the category of adjectives, which is a challenging part of speech for the CVA project, a word importunate has a "mysterious" nature. Depending on which context it is used the word importunate goes beyond the standard definition, describing various qualities. According to the protocols in the passage that was used for the project besides being defined as *demanding, pushy*, it is also realized as *sexually aggressive*. It would be interesting to see what results there would be, provided there were more people involved in the experiment. In this project the temporal representations were ignored. However, it would be helpful to represent them in the future. For instance, the process of Miss Wilkinson becoming importunate, on one hand, and Philip feeling discomfort and having a physical revulsion, on the other. It would also be interesting to see if the double occurrence of a word *sexual* in the passage had any correlation with the fact of people defining a word *importunate* as *sexually aggressive* frequently.

APPENDIX 1

A Demo of the Project

This file is the running demo of the project ("importunate.demo").

```
* (demo "/home/lingrad/petrova3/CSE727/importunate.demo" :av)

File /home/lingrad/petrova3/CSE727/importunate.demo is now the source of input.

  The demo will pause between commands, at that time press
  RETURN to continue, or ? to see a list of available commands

 CPU time : 0.00

* ;;;FILENAME: importunate.demo
;;;DATE: 10 May 2007
;;;PROGRAMMER: Nyurguyana Petrova
;;;
;;;INSTRUCTIONS on how to use this file:
;;;run SNePS:
;;;     run lisp (type acl)
;;;     :ld /projects/snwiz/bin/sneps
;;;change to package (type (sneps))
;;;at the SNePS prompt (:), type:
;;;(demo "importunate.demo" :av)
;;;
;;;=================================================================
```

```
;;;
 ;turn on full forward inferencing:
 ^(

--> defun broadcast-one-report (represent)
    (let (anysent)
      (do.chset (ch *OUTGOING-CHANNELS* anysent)
         (when (isopen.ch ch)
                 (setq anysent
                       (or (try-to-send-report represent ch)
                           anysent)))))
    nil)

broadcast-one-report

 CPU time : 0.00
*
 ;re-enter the "sneps" package:
 ^(

--> in-package sneps)

#<The sneps package>


 CPU time : 0.00
```

```
*

;;;================================================================

;;;clears knowledge base

(resetnet t)


Net reset


 CPU time : 0.00


*

;;;================================================================

;;;Defining some SNePS relations:

;;;member/class

;;;object/property


(define member class object property)


(member class object property)


 CPU time : 0.00


*

;;;================================================================

;;;object/propername

;;;object/location

;;;lex

;;;object1 rel object2
```

```
(define object propername location lex rel)

object is already defined.

(object propername location lex rel)

 CPU time : 0.00

*
;;;=============================================================
;;;cause/effect

(define cause effect)

effect is already defined.

(cause effect)

 CPU time : 0.00

*
;;;=============================================================
;;;agent/act/action/object

(define agent act action object)
```

```
act is already defined.

action is already defined.

object is already defined.


(agent act action object)


 CPU time : 0.00


*

;;;=================================================================

;;;PRIOR KNOWLEDGE

;;;=================================================================

;;;If someone is a person, and that person is in discomfort and in

physical revulsion,

;;;then he/she is not ok.


 (describe (assert forall $x

        &ant ((build member *x class (build lex "person"))

              (build object *x property (build lex "discomfort"))

              (build object *x property (build lex "physical revulsion")))

        cq (build object *x property (build lex "not-ok"))))


(m5! (forall v1)

 (&ant (p3 (object v1) (property (m3 (lex physical revulsion))))

  (p2 (object v1) (property (m2 (lex discomfort))))

  (p1 (class (m1 (lex person))) (member v1)))

 (cq (p4 (object v1) (property (m4 (lex not-ok))))))
```

```
(m5!)


 CPU time : 0.02


*
;;;================================================================
;;;If X is an unknown concept & if a person becomes X
;;;& if a second person is not OK because the first person became X,
;;;then the first person is pushy because that person is X.


;;;More Specifically:
;;;------------------
;;;If importunate is an unknown concept & if a person becomes importunate,
;;;& if a second person is not OK because the first person became
importunate,
;;;then the first person is pushy because that person is importunate.


(describe (assert forall ($x $p $q)
        &ant ((build object *x property (build lex "unknown"))
              (build object *p property *x)
              (build object *q property (build lex "not-ok"))
              (build cause (build object *p property *x)
                     effect (build object *q property (build lex "not-ok"))))
        cq (build cause (build object *p property *x)
                  effect (build object *p property (build lex "pushy")))))
```

```
(m8! (forall v4 v3 v2)
 (&ant
  (p8 (cause (p6 (object v3) (property v2)))
   (effect (p7 (object v4) (property (m4 (lex not-ok))))))
  (p7) (p6) (p5 (object v2) (property (m6 (lex unknown)))))
 (cq
  (p10 (cause (p6))
   (effect (p9 (object v3) (property (m7 (lex pushy))))))))

(m8!)

 CPU time : 0.00

*
;;;================================================================
;;;If X is an unknown concept
;;;& if a person becomes X
;;;& if a second person is not OK because the first person became X,
;;;then the first person is sexually aggressive because that person is X.

;;;More Specifically:
;;;------------------
;;;If importunate is an unknown concept,
;;;& if a person becomes importunate,
;;;& if a scond person is not OK because the first person became
importunate,
;;;then the first person is sexually aggressive because that person is
```

```
importunate.


(describe (assert forall ($x $p $q)
        &ant ((build object *x property (build lex "unknown"))
              (build object *p property *x)
              (build object *q property (build lex "not-ok"))
              (build cause (build object *p property *x)
                     effect (build object *q property (build lex "not-ok"))))
        cq (build cause (build object *p property *x)
                  effect (build object *p property (build lex "sexually aggressive")))))


(m10! (forall v7 v6 v5)
 (&ant
  (p14 (cause (p12 (object v6) (property v5)))
   (effect (p13 (object v7) (property (m4 (lex not-ok))))))
  (p13) (p12) (p11 (object v5) (property (m6 (lex unknown)))))
 (cq
  (p16 (cause (p12))
   (effect
    (p15 (object v6) (property (m9 (lex sexually aggressive))))))))


(m10!)


 CPU time : 0.00


*
;;;================================================================
```

```
;;;If p is f because p is x

;;;& if p is x

;;;then p is f


;;;More Specifically:

;;;------------------

;;;If person is pushy (or sexually aggressive) because person is

importunate,

;;;& if person is importunate,

;;;then person is pushy (or sexually aggressive).


(describe (assert forall ($p $f $x)

        &ant ((build cause (build object *p property *x)

                     effect (build object *p property *f))

              (build object *p property *x))

        cq (build object *p property *f)))


(m11! (forall v10 v9 v8)

 (&ant

  (p19 (cause (p17 (object v8) (property v10)))

   (effect (p18 (object v8) (property v9))))

  (p17))

 (cq (p18)))


(m11!)


 CPU time : 0.00
```

```
*
;;;=================================================================
;;;Forall x y [x makes advances towards y
;;;& x is older than y
;;;then x is pushy and sexually aggressive towards y]

(describe (assert forall ($x $y)
        &ant(build agent *x act (build action (build lex "makes advances towards") object *y))
        &ant(build object1 *x rel (build lex "is older than") object2 *y)
        cq(build object1 ((build object *x property (build lex "pushy"))
                          (build object *x property (build lex "sexually aggressive")))
                  rel (build lex "towards")
                  object2 *y)))

(m15! (forall v12 v11)
 (&ant
  (p22 (object1 v11) (object2 v12) (rel (m13 (lex is older than))))
  (p21
   (act (p20 (action (m12 (lex makes advances towards))) (object v12)))
   (agent v11)))
 (cq
  (p25
   (object1
    (p24 (object v11) (property (m9 (lex sexually aggressive))))
    (p23 (object v11) (property (m7 (lex pushy)))))
   (object2 v12) (rel (m14 (lex towards))))))
```

```
(m15!)

 CPU time : 0.00

*
;;;================================================================
;;;Forall x y [x is pushy & sexually aggressive towards y
;;;then y feels discomfort and physical revulsion
;;;because x is pushy and sexually aggressive towards y]

(describe (assert forall ($x $y)
        &ant((build object *x property (build lex "pushy"))
             (build object *x property (build lex "sexually aggressive"))
             (build object1 ((build object *x property (build lex "pushy"))
                             (build object *x property (build lex "sexually aggressive")))
                    rel (build lex "towards")
                    object2 *y))
        cq (build cause (build object1 ((build object *x property (build lex "pushy"))
                                        (build object *x property (build lex "sexually aggressive")))
                           rel (build lex "towards")
                           object2 *y)
                  effect ((build object *y property (build lex "discomfort"))
                          (build object *y property (build lex "physical revulsion"))))))

(m16! (forall v14 v13)
 (&ant
```

```
  (p28
   (object1
    (p27 (object v13) (property (m9 (lex sexually aggressive))))
    (p26 (object v13) (property (m7 (lex pushy)))))
   (object2 v14) (rel (m14 (lex towards))))
  (p27) (p26))
 (cq
  (p31 (cause (p28))
   (effect (p30 (object v14) (property (m3 (lex physical revulsion))))
    (p29 (object v14) (property (m2 (lex discomfort))))))))

(m16!)

 CPU time : 0.00

*
;;;================================================================
;;;Forall y x f [y feels discomfort and physical revulsion
;;;because x is pushy and sexually aggressive towards y
;;;& y feels discomfort and physical revulsion because x is f
;;;& f is unknown
;;;then
;;;Forall p1 [p1 is f
;;;then p1 is pushy and sexually aggressive]]

(describe (assert forall ($y $x $f)
        &ant (build cause (build object1 ((build object *x property (build lex "pushy"))
```

```
                                        (build object *x property (build lex "sexually aggressive")))
                                rel (build lex "towards")
                                object2 *y)
                    effect ((build object *y property (build lex "discomfort"))
                            (build object *y property (build lex "physical revulsion"))))
        &ant(build cause (build object *x property *f)
                   effect ((build object *y property (build lex "discomfort"))
                           (build object *y property (build lex "physical revulsion"))))
        &ant(build object *f property (build lex "unknown"))
        cq(build forall $p1
                   ant(build object *p1 property *f)
                   cq((build object *p1 property (build lex "pushy"))
                      (build object *p1 property (build lex "sexually aggressive"))))))


(m17! (forall v17 v16 v15)
 (&ant (p40 (object v17) (property (m6 (lex unknown))))
  (p39 (cause (p38 (object v16) (property v17)))
   (effect (p36 (object v15) (property (m3 (lex physical revulsion))))
    (p35 (object v15) (property (m2 (lex discomfort))))))
  (p37
   (cause
    (p34
     (object1
      (p33 (object v16) (property (m9 (lex sexually aggressive))))
      (p32 (object v16) (property (m7 (lex pushy)))))
     (object2 v15) (rel (m14 (lex towards)))))
   (effect (p36) (p35)))))
```

```
 (cq
  (p44 (forall v18) (ant (p41 (object v18) (property v17)))
   (cq (p43 (object v18) (property (m9)))
    (p42 (object v18) (property (m7)))))))

(m17!)

 CPU time : 0.01

*
;;;================================================================
;;;FROM TEXT
;;;================================================================
;;;Someone whose name is Philip

(describe (add object #philip propername "Philip"))

(m18! (object b1) (propername Philip))

(m18!)

 CPU time : 0.00

*

;;;================================================================
;;;Someone whose name is Wilkinson
```

```
(describe (add object #wilkinson propername "Wilkinson"))

(m19! (object b2) (propername Wilkinson))

(m19!)

 CPU time : 0.01

*
;;;================================================================
;;;Philip and Miss Wilkinson are persons.

(describe (add member (*philip *wilkinson) class (build lex "person")))

(m22! (class (m1 (lex person))) (member b2))
(m21! (class (m1)) (member b1))
(m20! (class (m1)) (member b2 b1))

(m22! m21! m20!)

 CPU time : 0.48

*
;;;================================================================
;;;Miss Wilkinson makes advances towards Philip.
```

```
(describe (add agent *wilkinson act

(build action (build lex "makes advances towards") object *philip)))


(m48!

 (act (m47 (action (m12 (lex makes advances towards))) (object b1)))

 (agent b2))


(m48!)


 CPU time : 0.00


*

;;;================================================================

;;;Miss Wilkinson is older than Philip.


(describe (add object1 *wilkinson rel (build lex "is older than") object2 *philip))


(m50!

 (object1 (m45 (object b2) (property (m9 (lex sexually aggressive))))

  (m43 (object b2) (property (m7 (lex pushy)))))

 (object2 b1) (rel (m14 (lex towards))))

(m49! (object1 b2) (object2 b1) (rel (m13 (lex is older than))))


(m50! m49!)


 CPU time : 0.01
```

```
*
;;;================================================================
;;;SNePS representation of the sentence from the text:
;;;As she becomes more importunate, though,
;;;Philip begins to feel discomfort and physical revulsion in her presence, <.>.
;;;
;;;From Maugham W. Somerset. (1915, 1991). Of Human Bondage.
;;;
;;;1). When Philip is in Miss Wilkinson's presence, he begins to feel
;;; discomfort and physical revulsion.
;;;2). Miss Wilkinson becomes (is) importunate.
;;;3). Philip feels discomfort and physical revulsion as Miss Wilkinson
;;; becomes (is) more importunate.

;;;================================================================
;;;When Philip is in Miss Wilkinson's presence, he begins to feel discomfort and physical revulsion.

(describe (add
        &ant ((build object *philip location #someplace)
              (build object *wilkinson location *someplace))
        cq ((build object *philip property (build lex "discomfort"))
            (build object *philip property (build lex "physical revulsion")))))

(m53!
 (&ant (m52 (location b3) (object b2)) (m51 (location b3) (object b1)))
 (cq (m29 (object b1) (property (m2 (lex discomfort))))
  (m25 (object b1) (property (m3 (lex physical revulsion))))))
```

```
(m53!)

 CPU time : 0.01

*
;;;=====================================================
;;;Philip and Miss Wilkinson are in the same place.

(describe (add object (*philip *wilkinson) location *someplace))

(m54! (location b3) (object b2 b1))
(m52! (location b3) (object b2))
(m51! (location b3) (object b1))
(m31! (object b1) (property (m4 (lex not-ok))))
(m29! (object b1) (property (m2 (lex discomfort))))
(m25! (object b1) (property (m3 (lex physical revulsion))))

(m54! m52! m51! m31! m29! m25!)

 CPU time : 0.16

*
;;;=============================================================
;;;Miss Wilkinson becomes (is) importunate.

(describe (add object *wilkinson property (build lex "importunate")))
```

```
(m56! (object b2) (property (m55 (lex importunate))))

(m56!)

 CPU time : 0.04

*
;;;================================================================
;;;As Miss Wilkinson becomes importunate, Philip feels discomfort and physical revulsion.

(describe (add cause (build object *wilkinson property (build lex "importunate"))
               effect ((build object *philip property (build lex "discomfort"))
                       (build object *philip property (build lex "physical revulsion")))))

(m57! (cause (m56! (object b2) (property (m55 (lex importunate)))))
 (effect (m29! (object b1) (property (m2 (lex discomfort))))
  (m25! (object b1) (property (m3 (lex physical revulsion))))))

(m57!)

 CPU time : 0.01

*
;;;================================================================
;;;Philip is not ok because Miss Wilkinson is importunate.
```

```
(describe (add cause (build object *wilkinson property (build lex "importunate"))
               effect (build object *philip property (build lex "not-ok"))))

(m58! (cause (m56! (object b2) (property (m55 (lex importunate)))))
 (effect (m31! (object b1) (property (m4 (lex not-ok))))))

(m58!)

 CPU time : 0.02

*
;;;================================================================

;;;Word (importunate) is unknown.

(describe (add object (build lex "importunate") property (build lex "unknown")))

(m63! (forall v18)
 (ant (p99 (object v18) (property (m55 (lex importunate)))))
 (cq (p43 (object v18) (property (m9 (lex sexually aggressive))))
  (p42 (object v18) (property (m7 (lex pushy))))))
(m62!
 (cause
  (m50!
   (object1 (m45! (object b2) (property (m9)))
    (m43! (object b2) (property (m7))))
   (object2 b1) (rel (m14 (lex towards)))))
```

```
 (effect (m29! (object b1) (property (m2 (lex discomfort))))
  (m25! (object b1) (property (m3 (lex physical revulsion))))))
(m61! (cause (m56! (object b2) (property (m55)))) (effect (m45!)))
(m60! (cause (m56!)) (effect (m43!)))
(m59! (object (m55)) (property (m6 (lex unknown))))

(m63! m62! m61! m60! m59! m56! m45! m43!)

 CPU time : 0.19

*
;;;================================================================
;;;Description of the built network:
(describe *nodes)

(m63! (forall v18)
 (ant (p99 (object v18) (property (m55 (lex importunate)))))
 (cq (p43 (object v18) (property (m9 (lex sexually aggressive))))
  (p42 (object v18) (property (m7 (lex pushy))))))
(m62!
 (cause
  (m50!
   (object1 (m45! (object b2) (property (m9)))
    (m43! (object b2) (property (m7))))
   (object2 b1) (rel (m14 (lex towards)))))
 (effect (m29! (object b1) (property (m2 (lex discomfort))))
  (m25! (object b1) (property (m3 (lex physical revulsion))))))
```

```
(m61! (cause (m56! (object b2) (property (m55)))) (effect (m45!)))
(m60! (cause (m56!)) (effect (m43!)))
(m59! (object (m55)) (property (m6 (lex unknown))))
(m58! (cause (m56!))
 (effect (m31! (object b1) (property (m4 (lex not-ok))))))
(m57! (cause (m56!)) (effect (m29!) (m25!)))
(m54! (location b3) (object b2 b1))
(m53!
 (&ant (m52! (location b3) (object b2))
  (m51! (location b3) (object b1)))
 (cq (m29!) (m25!)))
(m49! (object1 b2) (object2 b1) (rel (m13 (lex is older than))))
(m48!
 (act (m47 (action (m12 (lex makes advances towards))) (object b1)))
 (agent b2))
(m22! (class (m1 (lex person))) (member b2))
(m21! (class (m1)) (member b1))
(m20! (class (m1)) (member b2 b1))
(m19! (object b2) (propername Wilkinson))
(m18! (object b1) (propername Philip))
(m17! (forall v17 v16 v15)
 (&ant (p40 (object v17) (property (m6)))
  (p39 (cause (p38 (object v16) (property v17)))
   (effect (p36 (object v15) (property (m3)))
    (p35 (object v15) (property (m2)))))
  (p37
   (cause
```

```
    (p34
     (object1 (p33 (object v16) (property (m9)))
      (p32 (object v16) (property (m7))))
     (object2 v15) (rel (m14))))
   (effect (p36) (p35))))
 (cq
  (p44 (forall v18) (ant (p41 (object v18) (property v17)))
   (cq (p43) (p42)))))
(m16! (forall v14 v13)
 (&ant
  (p28
   (object1 (p27 (object v13) (property (m9)))
    (p26 (object v13) (property (m7))))
   (object2 v14) (rel (m14)))
  (p27) (p26))
 (cq
  (p31 (cause (p28))
   (effect (p30 (object v14) (property (m3)))
    (p29 (object v14) (property (m2)))))))
(m15! (forall v12 v11)
 (&ant (p22 (object1 v11) (object2 v12) (rel (m13)))
  (p21 (act (p20 (action (m12)) (object v12))) (agent v11)))
 (cq
  (p25
   (object1 (p24 (object v11) (property (m9)))
    (p23 (object v11) (property (m7))))
   (object2 v12) (rel (m14)))))
```

```
(m11! (forall v10 v9 v8)
 (&ant
  (p19 (cause (p17 (object v8) (property v10)))
   (effect (p18 (object v8) (property v9))))
  (p17))
 (cq (p18)))
(m10! (forall v7 v6 v5)
 (&ant
  (p14 (cause (p12 (object v6) (property v5)))
   (effect (p13 (object v7) (property (m4)))))
  (p13) (p12) (p11 (object v5) (property (m6))))
 (cq (p16 (cause (p12)) (effect (p15 (object v6) (property (m9)))))))
(m8! (forall v4 v3 v2)
 (&ant
  (p8 (cause (p6 (object v3) (property v2)))
   (effect (p7 (object v4) (property (m4)))))
  (p7) (p6) (p5 (object v2) (property (m6))))
 (cq (p10 (cause (p6)) (effect (p9 (object v3) (property (m7)))))))
(m5! (forall v1)
 (&ant (p3 (object v1) (property (m3)))
  (p2 (object v1) (property (m2))) (p1 (class (m1)) (member v1)))
 (cq (p4 (object v1) (property (m4)))))
(m63! p99 m62! m61! m60! m59! m58! m57! m56! m55 importunate m54! m53!
 m52! m51! b3 m50! m49! m48! m47 m45! m43! m31! m29! m25! m22! m21!
 m20! m19! Wilkinson b2 m18! Philip b1 m17! p44 p43 p42 p41 v18 p40 p39
 p38 p37 p36 p35 p34 p33 p32 v17 v16 v15 m16! p31 p30 p29 p28 p27 p26
 v14 v13 m15! p25 m14 towards p24 p23 p22 m13 is older than p21 p20 m12
```

```
 makes advances towards v12 v11 m11! p19 p18 p17 v10 v9 v8 m10! p16 p15
 m9 sexually aggressive p14 p13 p12 p11 v7 v6 v5 m8! p10 p9 m7 pushy p8
 p7 p6 p5 m6 unknown v4 v3 v2 m5! p4 m4 not-ok p3 m3 physical revulsion
 p2 m2 discomfort p1 m1 person v1)

 CPU time : 0.02

*
;;;================================================================
;;;The command 'show' is used to create diagrams of a built network
;;;It shows a node m57! and automatically saves it in a current directory with ps extension
(show m59 :file m59 :format ps)
;;;combined part of a network of nodes m59! m60! m61! m62! m63!
;;;showing definition inference
(show m59 m60 m61 m62 m63 :file definition_inference :format ps)
;;;a node m8!
;(show m8 :file m8 :format ps)
;;;a node m10!
;(show m10 :file m10 :format ps)
;;;the combined network of nodes m8! m10! m59! m60! m61! m62! m63!
(show m8 m10 m59 m60 m61 m62 m63 :file importunate_combined_nodes :format ps)
;;;a node m11!
;(show m11 :file m11 :format ps)

End of /home/lingrad/petrova3/CSE727/importunate.demo demonstration.
```

```
 CPU time : 1.01


*
```

APPENDIX 2

The Verbal Protocols

==========

Protocol 1(native speaker)

==========

Importunate - more open with one's feelings towards someone; more pushy.

The sentences from the passage that helped in figuring out a meaning of a word importunate:

* Though thirty-seven, Miss Wilkinson looks younger, and at
first fascinates and attracts Philip, who would like to think she can't be
more than twenty-six, <...>.> .

* <...> the English society <...> has rendered him vulnerable to the advances
of a woman <...>.

* <...> Philip, who would like to think she cant be more than twenty-six, only
six or seven years older than he is. (Protocol: Philip is younger than Miss Wilkinson)

* <...> she departs for governess position in Berlin. (Protocol: she is leaving that is why
she needs to develop the relationship quickly)

==========

Protocol 2 (native speaker)

==========

Importunate - sexually aggressie.

* <...> the English society <...> has rendered him vulnerable to the advances
of a woman <...>.

* <...> Philip begins to feel discomfort and physical revulsion in her presence <...>.

==========

Protocol 3 (non-native speaker)

==========

Importunate - opposite of hard to get; lustful, nymphomanic.

* Though thirty-seven, Miss Wilkinson looks younger <...>.

(Protocol: Miss Wilkinson is older than Philip)

* <...> at the age of eighteen, <...>. (Protocol: Philip is younger than Miss Wilkinson)

==========

Protocol 4 (non-native speaker)

==========

Importunate - pressing, advancing on someone;

* Though thirty-seven, Miss Wilkinson looks younger, and at first fascinates and attracts Philip, <...>.

* <...> the English society <...> has rendered him vulnerable to the advances of a woman <...>.

==========

Protocol 5 (native speaker)

==========

Importunate - interested in someone.

* Philip begins to feel discomfort and physical revulsion in her presence

==========

Protocol 6 (non-native speaker)

==========

First, the human subject defines a word as follows:

Importunate - flirtatious, open in one's sexual advances.

However, a sentence 'Miss Wilkinson <...>, and at first fascinates and attracts Philip <...>' makes the human subject change her/his mind. After reading the sentence the subject realizes that it is Philip who is attracted to Miss Wilkinson but not vice versa.
As a result the human subject provides a different definition of a word importunate:

Importunate - familiar, visible.

* Philip begins to feel discomfort and physical revulsion in her presence.

==========

Protocol 7 (native speaker)

==========

Importunate - acting older or seeming more mature, than prior, to an outside observer.

* Though thirty-seven, Miss Wilkinson looks younger, and at first fascinates and attracts Philip, who would like to think she can't be more than twenty-six, only six or seven years older than he is.
* As she becomes more importunate, though, Philip begins to feel discomfort and physical revulsion in her presence, and he is only too glad when she departs for governess position in Berlin.

;;;==========

Acknowledgments

I would like to thank Prof.Rapaport for his guidance throughout the whole process of this project. Also, I would like to thank graduate students and professors of the Department of Linguistics who participated in my experiment.

References

1 Rapaport, William J., Kibby, Michael W. (2002), "Contextual Vocabulary Acquisition: A Computational Theory and Educational Curriculum", in Nagib Callaos, Ana Breda, and Ma. Yolanda Fernandez J. (eds.), Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002; Orlando, FL)(Orlando: International Institute of Informatics and Systemics), Vol. II: Concepts and Applications of Systemics, Cybernetics, and Informatics I, pp. 261-266.

2 Rapaport, William J. (2003), "What Is the 'Context' for Contextual Vocabulary Acquisition?", in Peter P. Slezak (ed.), Proceedings of the 4th Joint International Conference on Cognitive Science/7th Australasian Society for Cognitive Science Conference (ICCS/ASCS-2003; Sydney, Australia) (Sydney: University of New South Wales), Vol. 2, pp. 547-552.

3 Rapaport, William J., Kibby, Michael W. (2002), "ROLE: Contextual Vocabulary Acquisition: From Algorithm to Curriculum", Grant proposal to NSF ROLE program.

4 Shapiro, Stuart C., Rapaport, William J. (1987), "SNePS Considered as a Fully Intensional Propositional Semantic Network", in Nick Cercone Gordon McCalla (eds.), The Knowledge Frontier: Essays in the Representation of Knowledge (New York: Springer-Verlag): 262-315.

5 Shapiro, Stuart C.; Rapaport, William J.; Kandefer, Michael; Johnson, Frances L.; Goldfain, Albert (2007), "Metacognition in SNePS", AI Magazine 28(1) (Spring): 17-31.