Contextual Vocabulary Acquisition of Verbs

Chris Becker

June 12, 2005

Abstract

This paper describes a computational implementation of an algorithm to define an ”unknown” verb based on the information in the context it appears in and any applicable background knowledge that the reader brings to bear. First, it describes some related research that has taken place and how it may be applied to the Contextual Vocabulary Acquisition (CVA) project. Next, it describes the process taken to enhance the current implementation of the CVA verb algorithm. This begins with an analysis of what information best conveys verb meaning, focusing on dictionary definitions as a model. It then moves on to how this information can be retrieved from the context of a verb and how it can be used to categorize the verb or determine its relationship to other ‘known’ actions. Next it describes the computational implementation of this algorithm and the results after applying it to a sequence of passages represented in the SNePS knowledge representation and reasoning system (Shapiro & Rapaport 1987). Lastly, it outlines some of the remaining issues left for future research in the CVA of verbs.

1 Introducing CVA

CVA is defined as the ‘active, deliberate acquisition of word meanings from text by reasoning from contextual clues, prior knowledge, language knowledge, and hypotheses developed from prior encounters with the word, but without external sources of help such as dictionaries or people’ (Rapaport & Kibby 2002). The goal of the CVA project is twofold: to develop and implement algorithms to define an unknown noun, verb, or adjective from the semantic representation of a sentence, and to develop a curriculum to enhance vocabulary learning. The results of the research in each of these areas is then cycled back to the other to further the development of each goal.

1.1 Previous CVA research

Previous research on CVA is made up of a very diverse body of literature including computational implementations and psychological studies. One of the goals of our research is to unify this large disparate body of literature and incorporate elements of both disciplines into our algorithms.

Over the past few decades, a number of similar approaches have been developed for computational CVA. Each of these systems has strengths that we are hoping to mirror and weaknesses that we are hoping to improve upon.

1.1.1 Granger (1977)

Richard H. Granger's system, called Foul-Up, is probably the earliest computational system designed to use contextual information to define an unknown word. The system works in conjunction with scripts, which represent stereotypical, domain-specific information and which serve as a set of expectations for a particular context. When the system comes across an unknown word, Foul-Up will attempt to place it in a slot representing a concept that the unknown word may fall under in the context of the script.

Foul-Up employs Roger Shank's theory of conceptual dependency to model events and define actions in terms of 10 primitive acts. The system uses the class inclusions of the verb's arguments as well as any prepositions relating them to match the verb to a specific primitive act. Using this information, the system can then match the unknown word to a known concept in the script.

Although the system is extremely limited by its reliance on domain-specific scripts, there are some elements that we have found to be worth borrowing for our own system, namely, verb categorization based on prepositions and argument classes.

1.1.2 Wiemer-Hastings (1998)

Peter Wiemer-Hastings et al.'s system, Camille, is capable of inferring verb meanings from context by inferring the semantic and syntactic constraints of the verb's arguments and predicate structure. It then uses these clues to place the unknown verb in a predefined ontology.

Many of the ideas that Wiemer-Hastings et al. describe have already been incorporated, or planned for in our algorithm. These include the following:

- Determining the syntactic frame (i.e. transitive, intransitive, ditransitive).
- Identifying reflexive construction.
- Determining that there is a syntactic subject (agent).
- Determining that there is a syntactic object.
- Determining that there is an indirect object.
- Determining the actors and object's semantic categories.
- Identifying 'with' construction (i.e., identifying instruments)

1.1.3 van Daalen-Kapteins and Elshout-Mohr (1981)

In addition to computational CVA, our work is built on numerous other research projects on the psychological aspects of CVA. Much research has been done on vocabulary learning, dating back to Werner and Kaplan, 1950. Some of the methodology employed by the CVA project stems from a more recent paper, by van Daalen-Kapteins and Elshout-Mohr, 1981.

The research done by Marianne Elshout-Mohr and Maartje M. van Daalen-Kapteijns provides a strong theoretical background for the CVA project. Their work centered on analyzing the ability of different readers to determine the meaning of an unknown word from context. The results of this research shed some light on the methods by which a 'good' reader can deduce word meanings from context.

Their findings showed that readers employ a model that provides a structure for information retrieval and modification of a current hypothesis in the task of word learning. As new information is added, good readers typically apply it to their hypothesized model of the word's meaning, while poor readers may simply discard the model and create a new one based on the newest information. With good readers, unless a serious contradiction occurs, the model remains relatively intact.

Summary

The large amount of research that has gone into CVA from various disciplines provides a strong theoretical foundation for the work we are doing. Psychological reseach on CVA has shown that there is a basis by which certain readers learn words with greater ease than others. The techniques employed may therefore be implemented in both computational systems and language learning curricula. Previous computational research provides plenty of evidence that we are indeed working with a tractable problem with a sound methodology.

2 Analyzing Verb Meaning

In order to develop an algorithm to retrieve a word's meaning from context it is first necessary to know what types of information to look for as well as where to find it. Since the ultimate goal of the computational portion of the CVA project is to have an algorithm output a well-formed definition of an unknown word, the best source to determine the components needed for such an output can be found in a dictionary. Below are some examples of definitions (retrieved from http://dictionary.reference.com/)for words that have been the focus of previous CVA research.

Augur: To predict, especially from signs or omens.

Comport: To conduct or behave (oneself) in a particular manner

Foist:

- **a.** To pass off as genuine, valuable, or worthy
- **b.** To impose (something or someone unwanted) upon another by coercion or trickery.
- **c.** To force onto another

Perambulate:

- **a.** To walk through.
- **b.** To walk about; roam or stroll.

cripple: To disable, damage, or impair the functioning

fold: (itr.)

- **a.** To close, especially for lack of financial success; fail.
- **b.** To give in; buckle.

proliferate:

- **a.** To grow or multiply by rapidly producing new tissue, parts, cells, or offspring.
- **b.** To increase or spread at a rapid rate.

pry:

a. To raise, move, or force open with a lever.

b. To obtain with effort or difficulty.

quail:

a. To shrink back in fear; cower.

b. To cause to fail in spirit or power; to quell; to crush; to subdue.

From the examples above, we can find seven major types of information present in dictionary definitions. These are listed in order of significance as follows:

1. Similar actions: All verbs are defined in terms of other verbs. These similar actions are usually potential sister classes or superclasses of the verb being defined.
2. Manner/Modifiers: These are components that constrain the similar action in such a way as to narrow the meaning down closer to that of the unknown verb. These often take the form of prepositions or prepositional phrases, representing elements such as:
   - **(a)** physical movement within the event space (e.g. 'through').
   - **(b)** Source and/or destination (e.g., 'upon').
   - **(c)** Properties of the agent (e.g., 'in fear').
   - **(d)** Properties of the object (e.g., 'as genuine').
   - **(e)** Specific method of performance or other corequisite acts (e.g., 'by rapidly producing new tissue', 'by coercion or trickery').
3. Effects/Results: This includes the intention of the action. (e.g., [to cause to] 'fail in spirit or power').
4. Causes/Enablement: This includes states of affairs that cause the action (e.g. 'lack of financial success') or other actions for which the unknown action is the result.
5. Instruments: The class membership of any physical object that the agent is using to accomplish the action. This is typically identifiable as any physical object appended to the predicate by 'with'.
6. Object/Indirect Object: The class membership of the syntactic object or indirect object of the verb.
7. Agent: The class membership of the agent of the action. Most of the time it is assumed that the agent is human, so this information is often not present in a definition.

Overall, the least common pieces of information present in dictionary definitions of verbs are the class of the object and agent. For the vast majority of verbs, these are inherited from the superclass of the verb(s) given in the definition. Only if the verb requires arguments of a narrower class than those of its superclass is this information is required.

An effective verb definition algorithm would need to return these seven elements in order to form a meaningful definition of an unknown verb. The next section will describe how some of these elements can be retrieved from a verb's context.

2.1 Identifying Components of Meaning in Context

The context surrounding a verb can be broken down into two parts: the immediate context and the wider context. The immediate context contains the verb and its arguments and any prepositions

linking them. The immediate context will also contain any additional modifiers of the action, such as adverbs or adverbial adjuncts that add information to the event occurring.

The relevant pieces of information that can be retrieved from the immediate context for the definition are: object and indirect object types; the agent type; any instruments; and any prepositions that determine motion, transfer, or the spatial relationships of the verb's arguments. Additionally, once we know all the arguments that the verb subcategorizes for, we can use that information to retrieve potential similar actions that subcategorize for the same items.

The wider context of the verb may span several sentences and include many ambiguous types of information. Often the wider context of the unknown verb may represent either the development of a description of a state or a causal chain of events which the unknown action is part of. It is difficult to develop a general set of rules that relate this information to the unknown action because the structure and content can vary greatly.

Sometimes the wider context will contain background knowledge or domain-specific rules of inference that may apply only within the context itself. These are often the types of things that a generic rule of inference cannot account for. Instead, it may be up to the parser (or the researcher representing the passage in SNePS) to identify this information. For example, consider the passage:

'Bob made sure he had his wallet on him. He wanted to ___ Maria dinner'

From this context, a reader may assume that the mention of 'wallet' in the first sentence presupposes that the unknown action has a requirement of money. This information however, would come from the readers background knowledge that wallets contain money and that a person needs to have their wallet in their immediate possession in order to use that money. In this case, the act of having money enables the unknown action to take place. Verbs that require money are typically those closely related to 'buy' or 'sell'. By reminding himself who has the money in this context, a reader may be able to narrow down the unknown verb to 'buy'.

Another type of context that may provide useful information is a conjoined event. An example of two types of conjoined events follows:

a. It rained in Dallas and snowed in New York.
b. I woke up and ate breakfast.

In (a), we may interpret that the two events occurred with respect to a single point in time, whereas in (b) we can infer that the two events occurred in sequence. In each of these examples, the information provided by one action puts a restriction on the possible state of the arguments or action in the second predicate.

In (a), we can tell that the two events share some components of meaning, namely 'precipitation'. In this context, the two actions subcategorize for the same class of object (a location) and both make use of the existential 'it'. These clues may presuppose some degree of simultaneity of the two events, which may therefore indicate a shared, higher-level verb class.

In (b), there are several clues that the two actions are not occurring simultaneously. The first is that both actions are being performed by the same agent. Most people would not assume that the agent is doing both actions at the same time, but doing one and then the other.

Presuming that a reader of passage (b) did not know the meaning of 'ate' but did know the meaning of 'woke', he might assume at the least that a requirement of 'ate' is that the agent must be awake. Further cultural knowledge might lead the reader to map the verb into his own knowledge of the sequence of events that may follow waking up.

Unfortunately the ease of retrieving a component of meaning of a verb from context is inversely proportional to the capacity of the information it carries. Similar actions are the most difficult to determine, due to the complexity of the inferences required and the ambiguity of the information that those inferences can be derived from. On the other hand, predicate structure and argument types can be acquired from a passage with great accuracy and detail.

2.2 Verb Categorization

If we can classify an unknown verb into one of a small set of primitive verb classes, we may be able to compromise between the need for a descriptive output and the absence of descriptive input.

Categorizing a verb based on a set of features is a simple task computationally and could also be converted to a simple curriculum. Students could easily be taught the meanings of a small set of primitive acts as well as a set of rules to determine whether a new verb is related in meaning to one or more of them. The same formula could be applied in the verb algorithm.

One framework that is applicable to CVA is Roger Shank's (1974) theory of conceptual dependency (CD). CD defines a set of 12 primitive acts from which, the theory claims, any action can be formed. These acts are defined as follows:

PTRANS: physical transfer of location of an object.

ATRANS: The abstract transfer of ownership, possession, or control of an object.

MTRANS: The mental transfer of information between agents.

MBUILD: The mental construction of a thought or new info between agents.

CONC: The conceptualizing or thinking about an idea by an animal.

ATTEND: The act of focusing attention of a sense organ toward an object.

GRASP: The act of grasping of an object by an actor for manipulation.

PROPEL: The applicaton of physical force to an object.

MOVE: The movement of a body part of an agent by that agent.

INGEST: The taking in of an object (food, air, water...) by an animal.

EXPEL: The expulsion of an object by an animal.

SPEAK: The act of producing sound, including non-communicative sounds.

The features used to determine whether an action is composed of a specific primitive act include predicate structure and thematic roles, both of which are easily accessible in the context of an unknown verb.

3 Implementation

The research on the computational implementation of CVA currently involves testing and enhancing the existing definition algorithms for nouns, verbs, and adjectives. Testing is done by creating a representation of a passage containing an 'unknown' word in the SNePS knowledge representation and reasoning system. The strength of SNePS lies in its 'reasoning' abilities, and thus provides a powerful framework on which to implement various theories of CVA.

A representation of a passage in SNePS will consist of three things: Background knowledge containing common sense facts relating to the 'known' concepts in the passage, a set of background rules embodying a set of common sense inferences possessed by a typical reader of the passage, and the representation of the passage itself. As each part of the passage is added to the SNePS network, inferences are generated based on any predefined rules and background knowledge, generating new propositions. The algorithm then searches the entire SNePS network and processes the relevant information needed to construct a possible definition of the unknown word.

3.1 Overview of Previous Implementations

3.2 CVA @ UB Timeline

1995 Karen Ehrlich creates algorithms for defining nouns and verbs. The verb algorithm retrieves the verb's predicate structure and causal/enablement information. Future work is to include in the output a primitive act of which the unknown verb is a type.

2002 Rajeev Sood enhances Ehrlich's verb algorithm to address information from direct and indirect objects and resolves some coding issues from her version. Future work includes developing a concept of basic level categories that verbs may fit into, identifying reflexive usage of verb, and keeping track of the state of the known 'world' before and after the action takes place in order to define results of the unknown action.

2002 Justin DelVecchio rewrites the verb algorithm. This version focuses on predicate structure, basic and top level categories of each predicate, and superclasses of the unknown verb. Future recommended work includes reviewing Hastings and Lytenins (1994) work on predicate information, implementing a formal structure for indirect objects, and reviewing Beth Levin's work on representing verbs in a hierarchy.

2003 Adel Ahmed works on a passage containing the verb 'cripple' in order to test the verb algorithm. Some modifications he suggests are including properties of the unknown verb and grouping common objects under a single superclass.

2003 Chris Becker works on a passage containing the verb 'pry'. Suggested modifications as a result of this work include the inclusion of information about instrument usage and prepositions that occur in the immediate context.

2003 Masashi Sato works on a passage containing the verb 'fold', and modifies the verb algorithm to return properties of the unknown verb. Future recommended modifications include allowing the verb algorithm to search for class membership and superclasses of the arguments of the unknown verb as well as retrieving synonyms of the unknown verb.

2003 Ananthakrishnan Ugrasenan Santha works on a passage containing the verb 'proliferate'. Future recommended modifications include extending the search for cause and effect in the network and processing sequences of actions.

2004 Chris Becker and Lunarso Sutanto rewrite the verb algorithm with emphasis on modular program design to allow future researchers to more easily make further enhancements. The algorithm returns information from the arguments of the unknown

verb as well as cause, effect, properties, and possible synonyms. Future recommended work includes implementing a classification system based on conceptual dependency primitive categories, addressing the role of morphology in identifying word meaning, testing for reflexivity, and implementing further rules to determine possible verb synonymy.

3.3 Current Implementation of the Verb Algorithm

This section will serve as a reference for future work on the verb algorithm and will outline the structure and describe the functions and flow of control that takes place. The overall design of the verb algorithm is implemented in two parts. The first part is a set of generic inference rules and background knowledge, written in SNePSUL. These rules are loaded into the system before any other information is added, and generate inferences as information is added from the context of the unknown word. The second part is the verb definition algorithm, which is written in Lisp. This part serves to retrieve and summarize relevant information stored in the SNePS network.

The verb algorithm is designed as a set of two modules that perform a specific set of tasks for each stage of execution. There are two main phases of execution: the data input and collection phase, and the data processing and output phase. The entire flow of control is maintained in the list *ProcessList* in the file defun_verb.cl. This list contains each function call in the order of execution. The function defineVerb must be called by the user to initiate the verb algorithm. This function takes the verb as an argument, defines the variable *OutputFrame* as the list of features to return values for, and executes the elements of *ProcessList*.

3.3.1 Data Collection

The goals for the data collection module of the verb algorithm were twofold. The first was to standardize the structure of certain representations pertinent to the meaning of the unknown verb. The second was to do this in such a way that modifications and additions could be made at a later time with a minimum amount of plodding through code and subsequent debugging.

The verb algorithm is currently coded to retrieve information from the following paths in the SNePS network.

Tables 3 and 4 list further information that is retrieved by the verb algorithm along with the path from which it retrieves this data. The function evaluateFindList in the data collection module then iterates through the list of paths and evaluates each path with the SNePS ‘find’ command. The output of this is stored in the associative list *Found-List*.

3.3.2 Data Processing

The next phase of execution uses the information stored in *Found-List* to determine the values for each field of the resultant definition frame for the unknown verb.

This phase is centered around the function assemble-data, which calls the appropriate functions to retrieve and assemble the data for the verb definition frame. These function calls are as follows:

Argument retrieved	Path from verb lex node
verb	(lex)
act	(action lex)
agent	(agent- act action lex)
object	(object- action lex)
indirect object	(indobj- action lex)
'from' argument	(from- action lex)
'to' argument	(to- action lex)
instrument	(instrument- act action lex)
'with' argument	(with- action lex)

Table 1: Arguments of the Verb. These paths are defined in the variable *ArgsToFind* in the data collection module. For each of these arguments, the algorithm retrieves the relations listed in table 2

Relation retrieved	Path from argument
lex	(lex-)
superclasses	(lex- superclass- subclass (kstar (compose superclass- subclass))))
class membership	(lex- class- member)
superordinate	(lex- superclass- subclass (kstar (compose superclass- subclass)) class- member)
properties	(lex- property- object)

Table 2: Relations of arguments of the verb. These paths are defined in the variable *ArgRelsToFind* in the data collection module.

1. Retrieve features that are directly connected to the unknown verb, namely properties and superclasses if present.
2. Retrieve the conceptual-dependency primitive category of the unknown verb.
3. Determine the list of similar actions. This list is compiled from the following:
   - **(a)** Equivalent actions defined in the SNePS network.
   - **(b)** Similar actions defined in the SNePS network.
   - **(c)** Actions in background knowledge that share the same conceptual dependency primitive category as the unknown verb.
   - **(d)** Other actions performed on the same object of the unknown verb in context or based on background knowledge.
   - **(e)** Other actions performed with the same instrument as the unknown verb in context or based on background knowledge.
   - **(f)** If none of the above contain any data, then retrieve other actions performed by the same agent in context or based on background knowledge.
   - **(g)** Sort the list formed by all of the above based on frequency of connectedness to the unknown verb in the SNePS network.
   - **(h)** Lastly, remove dissimilar actions from the above list.

Relation retrieved	**Path from verb lex node**
Facts that cause the action	(lex- cause- effect act action lex)
Facts that are the effect of the action	(lex- effect- cause act action lex)
Synonyms	(lex- synonym- synonym (kstar (compose synonym- synonym)) lex)
Similar actions	(lex- similar- similar (kstar (compose similar- similar)) lex)
Equivalent actions	(lex- equiv- equiv (kstar (compose equiv- equiv)) lex)
Actions performed by the same agent	(lex- action- act- agent agent- act action lex)
Actions performed on the same object	(lex- action- object object- action lex)
Actions performed by the object	(lex- action- act- agent object- action lex)
Actions performed with the same instrument	(lex- action- act- instrument instrument- act action lex)
Facts for which the action is the antecedent	(lex- action- act- ant- cq act action lex)
Facts for which the action is the consequent	(lex- action- act- cq- ant act action lex)
Actions for which the action is the cause	(lex- action- act- cause- effect act action lex)
Actions for which the action is the effect	(lex- action- act- effect- cause act action lex)

Table 3: Additional data retrieved by the verb algorithm. These paths are defined in the variable *MiscItemsToFind* in the data collection module.

Relation retrieved	**Path from verb**
Dissimilar actions	(lex- dissimilar- dissimilar (kstar (compose dissimilar- dissimilar)) lex)
CD primitive act	(lex- cd_cat- class lex)
Verbs with the same CD category	(lex- class- cd_cat cd_cat- class lex)

Table 4: Paths new to the current version of the verb algorithm. These paths are defined in the variable *MiscItemsToFind* in the data collection module.

4. Retrieve elements of the 'manner' in which the unknown action is performed. Currently this retrieves and returns the following:
   - **(a)** Whether there is transfer of location to or from the object.
   - **(b)** Whether there is an attribute that the agent is performing the action 'with'.
   - **(c)** Whether the agent is using an instrument.

5. Retrieve actions or single-node facts which caused the unknown action.

6. Retrieve actions or single-node facts which resulted from the unknown action.

7. Retrieve the transitivity of the unknown verb.
   - **(a)** This uses the result of a call to the function find-subcategorization which composes a list of the relations branching from the unknown action.
   - **(b)** The list returned from find-subcategorization is matched against lists which correspond to the types of transitivity (intransitive, transitive, ditransitive) and returns the correct one of the three.

8. Retrieve the most common superclass among all objects that the unknown verb occurs with.

9. Retrieve the most common superclass among all indirect objects that the unknown verb occurs with.
10. Retrieve the most common superclass among all instruments that the unknown verb occurs with.
11. Retrieve the most common superclass among all agents that the unknown verb occurs with.

One function used by many of the functions called by assemble-data, which I will describe in further detail, is the function return-basic-level-element. This function takes a list of base nodes in the SNePS network and determines the most common superclass among each of their class hierarchies. Since it is unlikely that each node in the input list has a single common superclass, the function returns a sorted list of classes based on their frequency of commonness among the hierarchies of all the base nodes.

This function is also used to determine the most common element among each of the sublists that make up the similar action list, which in effect returns the action with the greatest connectedness to the unknown action since each base node in this instance represents a different argument or relationship to the unknown verb.

Once the data are assembled into the associative list *Verb-Frame-List*, the function outputVerbFrame iterates through each feature-value pair and prints out those with a non-empty value. The output will contain any of the following features that the algorithm retrieved a value for:

- Similar actions in order from most probable to least probable
- Manner
- Properties of the verb
- Class membership of the verb
- Superclasses of the verb
- Superordinates of the verb
- Causes
- Effects
- Transitivity
- Object class
- Indirect object class
- Instrument class
- Agent class

The ultimate goal is to have the algorithm return a complete dictionary-like definition of an unknown word. This will require having both the components of meaning as well as a natural language engine to construct it. The current form of the output satisfies the first half of these requirements.

3.4 Additional Branches of Work

3.4.1 Verb categorization with conceptual dependency primitive acts

The implementation of verb categorization is done in two parts. The first stage uses a set of inference rules that match certain structures in the SNePS network. If it finds a particular structure containing

certain classes of arguments, it will assign the verb a CD category with the case frame: (class cd-cat). When the verb algorithm is executed, it retrieves this case frame and returns the CD category. It will also take this information and retrieve other verbs with the same CD category, which it will store in the list of similar actions.

An example of an inference rule that determines a verb's CD category is as follows:

```
(describe (assert forall ($A $B $C $D $E $F $G)
     &ant (
                (build agent *A act
                       (build action *B object *C to *D))
                (build member *A class *E)
                (build member *C class *F)
                (build member *D class *E)
                (build subclass *E superclass *entity)
                (build subclass *F superclass *physical_object)
          )
       cq (build class *B cd_cat *PTRANS)))
```

This rule basically states that if an entity performs an act with respect to a physical object 'to' another entity, then one component of meaning of the unknown verb must be 'the transfer of physical location of an object'.

Currently there are inference rules for the following primitive acts. These rules are stored in the knowledge file.

PTRANS: physical transfer of location of an object

ATRANS: The abstract transfer of ownership, possession, or control of an object

MTRANS: The mental transfer of information between agents

MBUILD: The mental construction of a thought or new info between agents

GRASP: The act of grasping of an object by an actor for manipulation

PROPEL: The applicaton of physical force to an object

MOVE: The movement of a body part of an agent by that agent

INGEST: The taking in of an object (food, air, water...) by an animal

3.4.2 Negation

Negation is a feature that has previously not been touched on in either the noun or verb algorithms. I've found, however, that it may potentially be useful, if only to prevent incorrect information from being returned as part of a definition. The implementation of this currently applies to negation of the (similar similar) case frame linking two actions.

First, the inference rule shown below is applied, which constructs a 'dissimilar' case frame linking the non-similar actions.

```
(describe (assert forall ($A $B)
        ant (build min 0 max 0 arg (build similar *A similar *B))
        cq  (build dissimilar *A dissimilar *B)))
```

The verb algorithm then retrieves instances of the (dissimilar dissimilar) path and removes the nodes it retrieves from the similar actions list.

3.4.3 Recommendations from Past Research

The following work that was recommended by past researchers has been implemented in the current version of the verb algorithm.

- Categorize the unknown verb based on a primitive act. (Ehrlich)
- List the arguments of the verb in the form of basic level categories. (Sood)
- Implement a formal structure for indirect objects (Del Vecchio)
- Return properties of the unknown verb. (Ahmed)
- Retrieve instrument usage. (Becker)
- Retrieve prepositions (Becker)
- Retrieve class membership and superclasses of the unknown verb (Sato)
- Retrieve additional representations of cause and effect (Santha)
- Classify the verb based on CD primitive acts (Becker & Sutanto)
- Construct rules to determine possible verb synonymy (Becker Sutanto)

The following issues brought up by past researchers have not yet been addressed. Below are my recommendations for future work in these areas.

Track the state of the world before and after the action (Sood)

The basic idea behind this is a good one; by determining which changes took place in the world when the action took place, we can infer what the action accomplished. The drawback of this, however, is that this information is not likely to occur in many contexts. This task should therefore be left in the hands of any researcher working on a context where this is applicable. It would most likely require a completely separate algorithm in order to be effective.

Processing sequences of actions using the 'snsequence' relation (Santha)

This would require augmenting the verb algorithm to retrieve instances of a structure such as:

```
(build action snsequence
        object1 (build act (build action *contaminate
                object *theproduct))
        object2 (build act (build action *proliferate))
        object3 (build act *attainlargenums)))
```

While not difficult to retrieve, processing this structure within the verb algorithm would be tedious, the final result of which would most likely be to presume that each successive act causes the next. If that is determined to be a valid assumption by a researcher using this structure, then it would probably be more efficient to create an inference rule that can match this structure and link each act with a cause effect case frame.

Address the role of morphology in CVA (Becker Sutanto)

Much further research would need to be done on the role of morphology in CVA before any aspect of it is implemented in the verb algorithm. Most likely the implementation would embody some knowledge base of morphemes and the primitive acts they represent. Additional case frames would need to be established to define the internal context of a word and the paths formed by these case frames would need to be added to the verb algorithm.

Identify reflexive usage of verbs (Becker Sutanto)

This would be fairly simple to implement. The function that would need to be added to the data processing module would retrieve the list of all known agents and objects; if the intersection of these lists is not null, then the action *may* be reflexive. If the intersection is not null and the set-difference of the two lists is null, then the action *must* be reflexive. The possible outputs of this function should be along the lines of one of the following:

- May be done to oneself.
- Can only be done to oneself.
- Is not done to oneself.

3.5 Support File Changes

The support files for CVA demos include the rels file, which defines all the necessary relations, and the paths file, which defines path-based relations. These files are called at the start of each demo in order to load a standard set of data. In addition to the above two, I have created a new 'knowledge' file to be loaded alongside them.

3.6 The Knowledge File

Outside of the verb algorithm, the biggest change I've made to the template of the CVA demo is the construction of a knowledge file. The purpose of this file is to contain generic background rules and knowledge for use in each CVA demo. This file current contains the following:

- Top level noun categories for use by inference rules. In order to be applicable, each physical entity defined in a demo file must be a subordinate of one of these top level categories.
- Conceptual dependency primitive categories. Currently there are multiple instances of each CD category which correspond to the separate rules that must fire to apply the category to a verb.
- A set of rules that define the structures and top level argument classes in order for a verb to be placed in a CD category.
- The following inference rules of verb similarity:
  - For each negation of a similar similar case frame, construct a dissimilar- dissimilar case frame.
  - For all agents, actions, and objects, actions performed on the same object by the same agent are similar.

- For all superclasses of each verb argument, actions performed on these same set of classes are similar.
- For all verbs and all CD categories, any two verbs that share the same CD category are similar.

3.7 Demos

Demos for the CVA project consist of a representation in SNePSUL code of a passage containing an 'unknown' word. The researcher constructing the demo must consider two important questions when determining what to represent:

- What clues would a good reader use from the context to hypothesize a meaning for the unknown word?
- What background knowledge is necessary for a reader to understand the unknown word in a passage?

The task of the researcher is therefore to accurately represent the appropriate information that a reader would need to understand the unknown word from context. Once the representations are entered into the SNePS network, the verb algorithm is called to attempt to define the unknown verb. If it successfully returns the elements of a usable definition of the unknown word, it serves as further evidence that the algorithm best represents the CVA process that readers undertake. If it does not retrieve the correct parts of a definition, then further work must be done to remedy the faults of either the demo or the algorithm.

The next three sections will list the passages that were represented in SNePS and the verb algorithm's output for each of them.

3.8 Contexts of 'perambulate'

This demo is contructed from six separate passages, some of which contain the verb 'perambulate', and others which contain elements of those contexts. These passages were entered into SNePS, and the verb algorithm executed after each was completed. From this sequence of examples, it is possible to see how the addition of new information caused the verb algorithm's hypothesized definition to change. (Note: for each context listed, the segment within brackets is the segment that was represented in SNePS. The remainder of the context is provided for the reader's convenience and in the event that future work may be done on a more detailed representation of any of these passages.)

Context 1

> [In the morning we rose to perambulate a city], which only history shews to have once flourished, [and surveyed the ruins of ancient magnificence] of which even the ruins cannot long be visible, unless some care be taken to preserve them; and where is the pleasure of preserving such mournful memorials? (Johnson, 1775)

Output 1

```
verb:
      lex:  perambulate;
```

```
      property:  unknown;
      similar action: (rose survey)
      transitivity: (transitive)
      object: (city)
      agent: (person)
```

Context 2

[We are going to attack a city and destroy it soon].

Output 2

```
verb:
      lex:  perambulate;
      property:  unknown;
      similar action: (unmake activity destroy attack)
      transitivity: (transitive)
      object: (city)
      agent: (person)
```

Context 3

[Nightly did the hereditary prince of the land perambulate the streets of his capital], disguised, well armed, alone, or with a single confidential attendant. (Motley, 1578)

Output 3

```
verb:
      lex:  perambulate;
      property:  unknown;
      similar action: (unmake activity destroy attack)
      transitivity: (transitive)
      object: (place)
      agent: (person)
```

Context 4

Between Paris in peace and Paris today the most striking difference is lack of population. Idle rich, the employees of the government, and tourists of all countries are missing. They leave a great emptiness. [When you walk the streets] you feel either that you are up very early, before any one is awake, or that you are in a boom town from which the boom has departed. (Davis, 1915)

Output 4

```
verb:
      lex:  perambulate;
      property:  unknown;
      similar action: (unmake activity destroy attack walk survey
                       perceive move)
      transitivity: (transitive)
      object: (place)
      agent: (person)
```

Context 5

> Ideas came to her chiefly when she was in motion. [She liked to perambulate the room] with a duster in her hand, with which she stopped to polish the backs of already lustrous books, musing and romancing as she did so. (Woolf, 1919)

Output 5

```
verb:
      lex:  perambulate;
      property:  unknown;
      similar action: (unmake activity destroy attack walk survey
                       perceive move)
      transitivity: (transitive)
      object: (place)
      agent: (person)
```

Context 6

> [Athena crossed the room to the other bed].

Output 6

```
verb:
      lex:  perambulate;
      property:  unknown;
      similar action: (unmake activity move crossed attack destroy
                       walk survey perceive)
      transitivity: (transitive)
      object: (place)
      agent: (person)
```

In each subsequent definition of ‘perambulate’, we can see how or whether the new information from the latest passage caused the verb algorithm to return a different output. Context 1 establishes the actions in the context as similar actions due to the fact that these are the only actions that SNePS ‘knows’ about.

Context 2 ‘teaches’ it a new pair of verbs that can occur in the context of the same arguments that are found in context 1 (i.e., ‘city’). This causes the algorithm to conclude that ‘unmake’ and ‘activity’ (superclasses of ‘attack’ and ‘destroy’) are the closest known actions to ‘perambulate’.

Context 3 presents ‘perambulate’ in the context of a new object (‘streets’). Now that the algorithm has seen both ‘city’ and ‘streets’ occur as objects, it determines that the most likely object is a ‘place’.

Context 4 presents what we know to be a possible synonym of perambulate (‘walk’) in a similar context. This causes the algorithm to add ‘move’ and ‘walk’ to the list of possible similar actions.

Context 5 reinforces the notion that an agent can perambulate a ‘place’. However it does not result in a change in definition.

Context 6 presents another possible synonym of perambulate (‘crossed’) in a similar context. Since the algorithm has now seen two other subclasses of the verb ‘move’ in similar contexts as perambulate, ‘move’ moves up the list of possible similar actions.

This demo serves to represent a possible situation that may occur in actual vocabulary learning, namely, the encoutering of examples that lead to false conclusions and others that reinforce the correct

conclusion of the meaning of an unknown word. Overall, however, it is most likely that a majority of contexts a reader may encounter will reinforce or narrow down the correct conclusion, thus leading to the correct result over time. In this respect, the fact that the algorithm correctly added emphasis to ‘move’ as a similar action in the end proves that the algorithm satisfactorily reproduces the results that a human vocabulary learner might with solely their knowledge of these contexts.

3.9 Conceptual Dependency Categories

This demo shows the use of the CD primitive act inference rules to categorize an unknown verb and the ability of the verb algorithm to retrieve them. The example sentences that were represented and the verb algorithm outputs are listed below:

Example 1

```
"Chris gave the book to john"

verb:
       lex:  gave;
       CD-CAT: (PTRANS(1a))
       transitivity: (transitive)
```

Example 2

```
"Chris drove from main street to maple"

verb:
       lex:  drove;
       CD-CAT: (PTRANS(3) PTRANS(2b))
       similar action: (gave)
       transitivity: (intransitive)
```

Example 3

```
"Chris went to the office"

verb:
       lex:  went;
       CD-CAT: (PTRANS(3))
       similar action: (drove)
       transitivity: (intransitive)
```

Example 4

```
"Chris showed John the story"

verb:
       lex:  showed;
       CD-CAT: (MTRANS(1b))
       similar action: (drove went gave)
       transitivity: (ditransitive)
```

Example 5

```
"Chris told John that Mary saw Bill"

verb:
       lex:  told;
       CD-CAT: (MTRANS(2))
       similar action: (showed)
       transitivity: (ditransitive)
```

Example 6

```
"Chris sent John the package"

verb:
       lex:  sent;
       CD-CAT: (PTRANS(4))
       similar action: (showed told)
       transitivity: (ditransitive)
```

Example 7

```
"The business offered stock to its shareholders"

verb:
       lex:  offered;
       CD-CAT: (MBUILD ATRANS)
       transitivity: (transitive)
```

Example 8

```
"Bill waved his hands"

verb:
       lex:  waved;
       CD-CAT: (MOVE(1))
       transitivity: (transitive)
```

Example 9

```
"I thought of a plan"

verb:
       lex:  thought;
       CD-CAT: (MBUILD ATRANS)
       similar action: (offered)
       transitivity: (transitive)
```

Example 10

```
"Chris ate a pomegranate."
```

```
verb:
       lex:  ate;
       CD-CAT: (INGEST)
       similar action: (sent thought drove went showed told gave)
       transitivity: (transitive)
```

Example 11

```
"Chris took the cup in his hands"

verb:
       lex:  took;
       CD-CAT: (GRASP)
       similar action: (thought ate went showed told sent gave drove)
       transitivity: (transitive)
```

Example 12

```
"John pushed the box onto the lift"
verb:
       lex:  pushed;
       CD-CAT: (PROPEL)
       transitivity: (transitive)
```

Although all the examples used in these tests are rigged to match a specific CD primitive act, this examples serve the purpose of illustrating what elements are required in the context for the inference rules to work. The following list summarizes which verbs were categorized with which primitive act:

1. gave: PTRANS
2. drove: PTRANS
3. went: PTRANS
4. showed: MTRANS
5. told: MTRANS
6. sent: PTRANS
7. offered: MBUILD ATRANS
8. waved: MOVE
9. thought: MBUILD ATRANS
10. ate: INGEST
11. took: GRASP
12. pushed: PROPEL

It should be noted that currently the inference rules for MBUILD and ATRANS match the same argument structure and types. Syntactic clues alone are not enough to differentiate the two.

For some of the above verbs it can be seen that the verb algorithm assigned related similar actions, while for others it did not. This is due to the fact that in the absence of any likely similar actions, the algorithm reverts to retrieving any known actions that the agent performed.

Some instances where it satisfactorily retrieved similar actions are in the cases of ‘showed’ and ‘told’, and ‘drove’ and ‘went’. Both of these resulted from the algorithm finding actions that shared the same CD category.

3.10 Evaluation of Past Demos

The following section lists some of the previous contexts that were worked on along with the output of running them with the current version of the verb algorithm.

3.10.1 A context of ‘augur’ (Becker 2004)

Context:

> Suddenly the tempest redoubled. The poor young woman could augur nothing favorable as she listened to the threatening heavens, the changes of which were interpreted in those credulous days according to the ideas or the habits of individuals.

Output:

```
verb:
       lex:  augur;
       property:  unknown;
       similar action: (interpret listen)
       cause: (bad omen heavens instance of change tempest listen
              interpret)
       transitivity: (transitive)
       object: (quality)
       agent: (human)
```

3.10.2 A context of ‘cripple’ (Ahmed 2003)

Context:

> Typhoon Vera killed or injured 218 people and crippled the seaport city of Keelung.

Output:

```
verb:
       lex:  cripple;
       property:  bad; unknown;
       similar action: (kill injure kill_or_injure)
       transitivity: (transitive)
```

3.10.3 A context of ‘fold’ (Sato 2003)

Context:

> During the early 1930s, many small businesses folded(fold: to go out of business) OR large corporations bought them up.

Output:

```
verb:
       lex:  fold;
       property:  unknown; destructive to business;
       similar action: (go out of business)
       transitivity: (intransitive)
       agent: (organization)
```

3.10.4 A context of ‘proliferate’ (Santha 2003)

Context:

Postfarm food processing, storage, and improper handling and cooking are major contributors to the chain of events that allows the pathogen to contaminate the product, proliferate on or in the food, and attain the large numbers that cause disease.

Output:

```
verb:
       lex:  proliferate;
       similar action: (multiply)
       cause: (contaminate)
       effect: (attain)
       transitivity: (intransitive)
```

3.10.5 A context of ‘pry’ (Becker 2003)

Context:

If everything is unbolted, unscrewed, and unattached, pry the door panel from the door using a wide, thin screwdriver”

Output:

```
 verb:
       lex:  pry;
 manner:
       movement:  from object;
       using:  instrument;
       cause: (Everything being disconnected)
       effect: (Something is removed)
       transitivity: (transitive)
       instrument: (screwdriver tool)
```

3.10.6 A context of ‘quail’ (Sutanto 2003)

Context:

The dark figure streaming with fire raced towards them. The orcs yelled and poured over the stone gangways. Then Boromir raised his horn and blew. Loud the challenge rang

and bellowed, like the shout of many throats under the cavernous roof. For a moment the orcs *quailed* and the fiery shadow halted. Then the echoes died as suddenly as a flame blown out by a dark wind, and the enemy advanced again.

Output:

```
verb:
       lex:  quail;
       similar action: (yell advance)
       transitivity: (intransitive)
```

4 Remaining Issues and Future Work

There are a number of potential areas of future work in the CVA project, especially in areas with respect to verbs. Some general trends I'd like to see in the future are a greater emphasis on studying and representing multiple contexts of the same verb to assess how the information from each can enhance or refute assumptions from the other, and further work creating a general set of inference rules to classify verbs based on features of their arguments. In the following sections, I will list some of the specific tasks that I recommend for future researchers to work on.

4.1 Other Categorization Schemas

One branch of future work on the verb algorithm would be to implement another method of verb categorization. Beth Levin's verb classes would provide a much greater level of detail than that of CD, but would take much more work to implement.

Another possibility would be to use Webster and Marcus's (1989) theory of feature sets to build a verb hierarchy in which new actions can be placed. Much of the information that is required by their system is based on thematic roles, which matches very closely to the case frames we use. Examples of some of the features used include 'takes an object', 'takes an agent', and 'takes a theme'.

During execution, the algorithm would place the verb in some position in the verb hierarchy based on the features present. As further contexts of the verb are added, the verb may move to a different position in the hierarchy, if it finds that any of its established features are optional, or if new optional features must be added.

Creating an implementation of this system should not be difficult since all of the features of the verb are already present in the SNePS network. These features can be retrieved by the verb algorithm and matched to a predefined hierarchy (e.g., via an association list with the features list as the key and the verb class as the value).

Further refinement can be made by computing the 'required' and 'optional' features based on evidence from multiple contexts. Using the intersection function on the list of all of the unknown verb's arguments would produce the list of required features, while the 'set-difference' of this required features list and the complete list of arguments of the unknown verb would produce the list of optional features. The different optional and required features lists could then be used to match the unknown verb to a more specific verb category.

4.2 Further Work on Background Knowledge and Inference Rules

Further work on CVA will most likely focus on enhancing the set of general rules and background knowledge that may be employed to infer data pertinent to defining an unknown word.

In terms of background knowledge, further work on categorization methods will undoubtedly require augmenting the number of top-level categories with which to define them.

As a note for future researchers, there have been some issues I have encountered while implementing inference rules and background knowledge. First, in order to incorporate rules and background knowledge in such a way that they will take effect, the rules must be 'asserted' first, and the propositions 'added' later. The inference rules are not applied if any of the necessary background knowledge was 'asserted'.

Another factor that someone may encounter a problem with during the construction of further inference rules is unique variable binding. This feature prevents two variables from pointing to the same node. An example where this might cause a problem is with a rule of subcategorization that is not built to handle reflexivity. That is, if the rule is meant to match each of the verb's arguments, but two of those arguments are actually the same node, then the rule will not register a match. This can be worked around by developing multiple rules, one of which handles the reflexive cases with a single variable, and the other which handles all other cases.

Lastly, I should make a recommendation that since the amount of background knowledge and inference rules can potentially be expanded a great deal, it would be advisable to develop a new form of organization for this data (e.g., a background knowledge file and an inference rule file). Additionally, within each of these files, some form of organization must be maintained (e.g., modularize by categorization theory) .

4.3 Representing Prepositions

Another issue that has remained open-ended for some time now is the representation of prepositions. Two contending formats of representing prepositions in SNePS are as follows:

- represent the preposition as a node with a 'prep' arc, and represent the argument of the preposition as a separate node with an 'arg' arc.
- represent the argument of the preposition as a node with an arc labeled to represent the actual preposition.

Each of these has its advantages and disadvantages. The former would appear to be more flexible, while the latter would make it easier to organize in the verb algorithm. Currently I have chosen to implement the second method for the following reasons:

- the verb algorithm retrieves information based on paths, not on the values at the end of the path.
- Implementing the 'prep' arc would require that another level of processing take place to separate the arguments for each type of preposition.
- Either way it will still be necessary to have a list of 'known' prepositions along with methods of processing each of them, so the advantage of flexibility that the first method has is somewhat nullified.

- Prepositions are a closed class of words. The flexibility that the first method offers is not really needed unless we expand into multilingual CVA, so there is only need for a finite number of new relations to be defined.

Further work on prepositions will be needed if more specific verb categorization schemes are to be implemented. Currently several of the CD category rules of inference utilize a 'to' arc to identify movement between entities. This can eventually be expanded for different classes of prepositions such as those representing spatial relationships (e.g. 'in', 'on', 'at'), or corequisite properties or events (e.g. 'with', 'by', 'as', 'during').

4.4 Looking Ahead to Verb Algorithm 4.0

With the current version (3.1) of the verb algorithm completed, its functionality is now comparable to that of the noun algorithm, which has for a long time been several stages ahead.

The next rendition of these algorithms should probably work on combining the two into one (and then seeing if an adjective algorithm can be incorporated). In terms of design and ease of modification, the current version of the verb algorithm would probably be the best to build from. The only modifications necessary would be to define the set of paths that the algorithm must search through to define an unknown noun, and to define the list of features of the noun definition frame. The only functions that would need to be written (or moved over from the current noun algorithm) are those which populate each of the noun definition frame's features.

Once the complete functionality needed for the unified definition algorithm is hammered out, the next step would be to convert the algorithm into SNeRE, integrating it with the cognitive agent of the SNePS system.

4.5 Conclusion

The current implementation of the verb algorithm now accomplishes what many previous researchers of the CVA of verbs have wanted it to do. Based on the results of the verb demos it appears that the algorithm is sound. There is still much future work to be done in CVA, however. Below, I outline the main stages of the CVA process and the implementation needed to bring it to completion:

CVA Process	**Possible Implementation**
1. Read the passage.	A parser and grammar (ATN or LKB)
2. Retrieve background knowledge associated with each element of the passage.	Wordnet, Cyc
3. Combine known background knowledge and information from the passage.	Context-specific inference rules, SNePS
4. Generate a set of generic inferences to determine the word's relationship to other elements in the network.	General inference rules, SNePS
5. Determine which information is relevant to the meaning of the unknown word.	Word definition algorithm, SNeRE
6. Formulate a grammatical definition.	SNeRE, SNaLPS, ATN, LKB.

References

Carbonell, Jaime G. 1979. "Towards a Self-Extending Parser", *Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics (University of California at San Diego)* (Morristown, NJ: Association for Computational Linguistics): 3-7.

Hastings, Peter M., and Lytinen, Steven L. 1994. "Objects, Actions, Nouns, and Verbs", *Proceedings of the 16th Annual Conference of the Cognitive Science Society*. (Hillsdale, NJ: Lawrence Erlbaum Associates): 397-402.

Granger, Richard H. 1977. "Foul-Up: a Program that Figures Out Meanings of Words from Context", *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI-77, MIT)* (Los Altos, CA: William Kaufmann): 67-68.

Levin, Beth. 1993. *English Verb Classes and Alternations*. (Chicago, Chicago University Press)

Rapaport, William J., and Ehrlich, Karen. 2000. "A Computational Theory of Vocabulary Acquisition", in Lucja M. Iwanska and Stuart C. Shapiro (eds.), *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. (Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press): 347-375.

Rapaport, William J., and Kibby, Michael W. 2002. ”ROLE: Contextual Vocabulary Acquisition: From Algorithm to Curriculum”,

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

Schank, Roger C., and Rieger, Charles J., III. 1974. ”Inference and the Computer Understanding of Natural Language”, *Artificial Intelligence* 5: 373-412.

van Daalen-Kapteijns, M.M., and Elshout-Mohr, M. 1981. ”The Acquisition of Word Meanings as a Cognitive Learning Process”, *Journal of Verbal Learning and Verbal Behavior* 20: 386-399.

Webster, Mort, and Marcus, Mitch. 1989. ”Automatic Acquisition of the Lexical Semantics of Verbs from Sentence Frames”, *Proceedings of the 27th Conference on Association for Computational Linguistics*. (Vancouver, British Columbia, Canada): (Association for Computational Linguistics): 177-184.

Werner, Heinz, Kaplan, Edith 1950. ”Development of Word Meaning through Verbal Context: An Experimental Study”. *Journal of Psychology* 29: 251-257.

Wiemer-Hastings, Peter; Graesser, Arthur C. and Wiemer-Hastings, Katja. 1998. ”Inferring the Meaning of Verbs from Context”, in *Proceedings of the 20th Annual Conference of the Cognitive Science Society*.

CVA Student Progress Reports. Available at: http://www.cse.buffalo.edu/∼rapaport/CVA/cvapapers.html

Becker, Chris 2003 ”Contextual Vocabulary Acquisition: On a Representation of Everything but the Word ‘Pry’”.

Becker, Chris 2004 ”Contextual Vocabulary Acquisition; Contextual Information in Verb Contexts: From Analysis to Algorithm”.

Del Vecchio, Justin M. 2002 ”A Contextual Vocabulary Acquisition Algorithm for Verbs, Implemented in SNePS”.

Sood, Rajeev 2002 ”Verb Acquisition in Computational Cognitive Agents”.

Sutanto, Lunarso 2003 ”Inferring the Meaning of ‘Quail’”.

Sutanto, Lunarso, Becker, Chris 2004 ”Verb Algorithm Revision Final Report”.

Ugrasenan Santha, Ananthakrishnan 2003 ”Deducing Verb Meanings from Context”.

Sources of CVA Contexts.
Project Gutenberg. http://www.gutenberg.net/

Davis, Richard Harding. 1915. *With the Allies*.
http://www.greatwardifferent.com/Great_War/Paris_at_War/With_the_Allies_01.htm

Johnson, Samuel. 1775. *A Journey to the Western Isles of Scotland*.
http://www.gutenberg.net/etext00/jwsct10.txt

Motley, John Lothrop. 1578. *The Rise of the Dutch Republic*.
http://www.gutenberg.net/etext04/jm31v10.txt

Woolf, Virginia. 1919. *Night and Day*. http://www.gutenberg.net/etext98/niday10h.htm