Semantic Representation and Analysis (SRA) and Its Application in Conversation-Based Intelligent Tutoring Systems (CbITS)

Xiangen Hu | The University of Memphis and Central China Normal University

Zhiqiang Cai and Andrew Olney | The University of Memphis

ABSTRACT

This chapter provides an overview of semantic spaces and introduces a mathematical framework called Semantic Representation and Analysis (SRA) developed by the authors. It then demonstrates how such a framework enables efficient conversationbased learning environments, using AutoTutor as an example of conversation-based learning environments. The chapter uses an alternative model of intelligent tutoring systems and argues that SRA is most useful in knowledge representation of the domain and in enhancement of evaluation methods in conversation-based learning environments.

It has long been a dream of learning scientists to produce computerized learning environments that enable computers to teach human deep knowledge similar to skilled human tutors (du Boulay & Luckin, 2001). According to explanation-based constructivist theories of learning (Aleven & Koedinger, 2002; VanLehn, Jones, & Chi, 1992), an effective and deep learning environment needs to be able to guide the learner to generate explanations and functional procedures. To provide this kind of instruction, researchers have been trying to develop intelligent tutoring systems (ITS) that adaptively respond to the learner's actions and language inputs (Anderson, Corbett, Koedinger, & Pelletier, 1995). One of the challenging tasks in such systems is to infer what the learner knows from natural language input.

Over the last few decades, substantial research has investigated technologies such as text mining and natural language processing (NLP) to address this challenge. These technologies are the basis for producing so-called semantic spaces. A semantic space represents the semantics of language entities (words, phrases, sentences, etc.) and the similarity as distances between them (Dumais, Furnas, Landauer, Deerwester, & Harshman, 1988; Landauer, Foltz, & Laham, 1998; Riordan & Jones, 2011). Although it was originally developed for entirely different purposes such as information retrieval (IR; Furnas et al., 1988; Landauer et al., 1998), it has been used for assessment of learning, cognitive modeling, and a variety of other learning tasks (Landauer, 2003). One of the most wellknown examples of a semantic space is latent semantic analysis (LSA) (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990; Landauer, 2006), and it has been used widely in advanced learning environments such as conversation-based ITS for over 20 years (Nye, Graesser, & Hu, 2014). Semantic encoding and decoding algorithms similar to LSA have been developed and applied, such as latent Dirichlet allocation (LDA; Blei, Ng, & Jordan, 2003; Hoffman, Bach, & Blei, 2010; Shams & Baraani-Dastjerdi, 2017), GloVe (Pennington, Socher, & Manning, 2014), and Word2Vec (Goldberg & Levy, 2014; Rong, 2014; Yu & Dredze, 2014). Semantic spaces have primarily been used to measure learning and to provide a cognitive model of the learner. The advances in research, development, and application of semantic spaces made this dream of learning scientists a possibility.

In this chapter, we will first provide a brief overview of semantic spaces and introduce a mathematical framework called Semantic Representation and Analysis (SRA) developed by the authors (Hu et al., 2014). Then we will demonstrate how such a framework would enable efficient conversation-based learning environments. Our example of a conversation-based learning environment will be AutoTutor (Graesser, Dowell, & Clewley, 2017; Graesser, Hu, & Person, 2001; Nye, Graesser, & Hu, 2014; Nye, Graesser, Hu, & Cai, 2014).

SEMANTIC REPRESENTATION AND ANALYSIS (SRA)

Semantic Representation and Analysis (SRA) is a canonical class of models that was first introduced by the authors when they attempted to measure differences between different semantic encoding and decoding methods (Hu, Cai, Graesser, & Ventura, 2005). It was then used as a general framework in extracting, representing, and applying semantic analysis in a conversation-based ITS (Hu et al., 2014). SRA can be understood as a class of methods in computational linguistics analysis (Gabrilovich & Markovitch, 2007; Landauer, McNamara, Dennis, & Kintsch, 2013) that specialized in encoding and decoding semantics from a large body of texts.

A Brief Overview of Vector-Based Semantic Models There are two ways to categorize semantic models. When considering the original definition of semantics, a class of semantic models is called distributional models where semantic meanings of language entities are defined by the ways they have associated with other language entities (Harris, 1954; Riordan & Jones, 2011). As an intuitive example, in a distributional model, the semantic meaning of a word is defined by how the word appears in language environments (such as paragraphs) with other words in a selected body of text data. Another way to categorize semantic models is based on the resulting mathematical form of the semantic representation. Vector-based model is a class of semantic models in which the final semantic representation is in the form of a vector. Vector representation of the semantics of a term can be obtained by human experts. For example, the word association norms (Nelson, McEvoy, & Schreiber, 2004), in which each term is associated with a small collection of other terms, is an example of a vector-based model where the dimension of the vector is large, but each term vector only has a small number of nonzero elements. In general, vector representation can be explicit or latent with the similar computations (Hu, Cai, Wiemer-Hastings, Graesser, & McNamara, 2007).

The basic vector-based model has been used for decades in information retrieval (Salton, Wong, & Yang, 1975). In their original model, words are orthogonal, so semantic similarity judgments are too brittle to be of use in learning environments. However, the basic model was later extended by latent semantic indexing (LSI) for information retrieval (Deerwester et al., 1990). The breakthrough provided by LSI is a solution to the synonymy problem, in which multiple words can express similar meaning. In the basic vector-based model, distinct words with similar meaning are kept distinct, but LSI gives them equivalent or near equivalent meanings. LSI's solution to the synonymy problem made it attractive to a variety of researchers outside of information retrieval (Graesser et al., 2001; Hu, Cai, Franceschetti, et al., 2003; Hu, Cai, Louwerse, Olney, Penumatsa, & Graesser, 2003; Landauer et al., 1998; Olney & Cai, 2005a, 2005b; Wolfe et al., 1998).

While the meaning of words seems to be very well encoded in LSI (currently often referred to as LSA, for latent semantic analysis), the meaning of each dimension of an LSI space is unknown. Latent Dirichlet allocation (LDA; Blei et al., 2003) provided an acceptable solution to this. LDA assumes (1) each document in a given corpus is a mixture of a given set of topics with a probability distribution over the topics; (2) each word appears in a topic with a certain probability, and (3) each word in a given document is drawn from a topic with the given topic probability and the word probability. The topic probability distribution of each document and the word probability distribution of each word is determined by maximizing the likelihood of the given corpus. Once determined, each word has a probability value on each topic and thus gets a vector representation as weights to the given topics. What made researchers excited about LDA is that the words with the highest probabilities in each topic often form a good interpretation of the topic. In other words, researchers can often see what a topic is by looking at the words ranked at the top by probability. Once an LDA model is built (i.e., the document topic probabilities and the topic word probabilities are found), any text can be represented by a vector known as "topic proportion scores," indicating how much each known topic is contained in a text. Similar to LSI, such vector representation can be used to compute semantic similarities between texts (Cai, Li, Hu, & Graesser, 2016). Word2Vec is a more recent way of generating word vector representations. Word2Vec uses two-layer neural networks to train vectors from a given corpus. The basic idea is to position word vectors in a space so words that share common contexts are located closely (Goldberg & Levy, 2014).

Similar to LSA (Deerwester et al., 1990; Landauer et al., 1998), most of the other methods such as LDA (Blei et al., 2003; Hoffman et al., 2010; Shams & Baraani-Dastjerdi, 2017), and Word2Vec (Goldberg & Levy, 2014; Rong, 2014; Yu & Dredze, 2014) are examples of vector space models. In the next section, we outline the steps to create vector-based semantic spaces. The Complexity of Creating Vector-Based Semantic Models Conceptually, representing the semantic meaning of a term in the form of vectors is intuitive, at least to those who think computationally. The goal of creating an alternative vector (mathematical) representation of the semantic of the original language entities is to make it possible for computers to compute semantic relations between the relevant language entities. There are several available computational/statistical techniques to achieve this goal. As an example, LSA uses singular value decomposition (SVD; Golub & Kahan, 1965). SVD is a technique that creates an approximation of the original word by document matrix. Other vector space models (such as LDA, Word2Vec) most often used recently are results of sophisticated statistical techniques that represent the similarity between collections of words as vectors "distance" in multidimensional space (Manning & Schütze, 1999). The process begins by collecting text into a corpus. A matrix is created from the corpus, having one row for each unique word in the corpus and one column for each document or paragraph. The cells of the matrix consist of a simple count of the number of times word i appeared in document j. Since each document only contains a small number of word types, the matrix is often highly sparse. Weightings are often applied to the cells that take into account the frequency of word i in document j and the frequency of word i across all documents, such that distinctive words that appear infrequently are given the most weight. Two collections of words of arbitrary size are compared by creating two vectors. Each word is associated with a row vector in the matrix, and the vector of a collection is simply the sum of all the row vectors of words in that collection. Vectors are compared geometrically by the cosine of the angle between them. This geometric interpretation is likely the reason that vector space models have become so popular: the interpretation is simple, clean, and elegant (Hu et al., 2007).

To explicitly outline the above-described process, we use the example steps of LSA to offer a taste of the complexity of the process. There are seven steps: ED: There are 9 steps listed:

1. Selection of a domain. This is determined by the purpose of semantic space. For example, a semantic space may be used as a semantic engine for a tutoring system of a specific domain (Franceschetti et al., 2001; Olde, Franceschetti, Graesser, & Karnavat, 2002) or for grading essays in a given subject (Foltz, Laham, & Landauer, 1999).
2. Selection of corpora. This step involves collecting and selecting relevant texts for the given domain. For example, if the domain is physics, there is a choice of textbooks, research articles, or both.
3. Processing of raw material. It is possible that collected material contains not only text but also graphics (a bitmap of a scanned document). Even if the materials are texts, there may be strings that are tags and attributes (for example, in HTML files). This step also includes inserting a proper sentence or paragraph markers. In some cases, it is at this step that the researcher may need to decide the size of the document.
4. Obtain word-document frequency matrix A. at this step, one needs to decide the value of each entry of the matrix as a function of global weight gi (weight of the word i) and local weight lj (weight of the document j) and the frequency of word i in document j: fij. It usually takes the form of (A)ji = giljfij. Notice that gi, lj, and fij can take different forms.
5. Decomposition of the word-document matrix to represent words by vectors. There are different ways one could process the matrix. Even SVD is the most popular method used, but one could use others to represent words as vectors.
6. Dimensional reduction. This is the step to determine how many dimen sions are enough for a vector representation. Some scholars report that 300 is the best for some applications (Landauer et al., 1998). However, there are cases in which higher dimensions are used.
7. Processing of the vectors. This is the step to decide (1) if the first dimension of the word vector needs to be removed, and (2) if the dimensions need to be weighted (Hu, Cai, Franceschetti, et al., 2003). After the seven steps, each word is represented as an n-dimensional vector with n being a number chosen at step 7: There are also different ways to use the vectors to compute similarity.
8. Similarity computation. Cosine is the default similarity measure for LSA. Researchers have also suggested that other similarity measures can be used.
9. Use of the similarity value. In the case of comparing similarity values between documents, one could use the value by itself, or one could use the value and consider the size (number of words) of the document.

Other semantic spaces such as Hyperspace Analogue to Language (HAL; Burgess, 1998) and non-latent similarity (NLS) algorithm (Cai et al., 2004) are only different from LSA in some of the steps (such as step 5, for example). The first three of the above-listed steps are essential for generating any semantic spaces. To make sure a given semantic space is truly domain-specific, some special domain and corpora selection algorithms need to be used. For example, the so-called seeding method suggested by Cai et al. (2018) makes them intuitive and efficient.

The seeding method starts with a small seed corpus representing the core of a domain. By comparing with a general reference corpus, the domain keywords are extracted and each keyword is assigned a "keyness" value. With these keyness values, documents in a large corpus (e.g., Wikipedia articles) are ranked by the total sum of keyness of words in each document. Top documents are then be selected into the domain-specific corpus.

In general, there are many parameters that need to be determined for any semantic space. One of the challenges is to select the best set of parameters for a given application. For example, there are possible alternatives for each of the nine steps outlined above. With so many choices at each of the steps, the resulting number of semantic spaces is combinatorial. This makes it impractical to evaluate and select an appropriate semantic space for a given application. The general framework of vector-based semantic space called semantic representation and analysis (SRA; Hu et al., 2014) was introduced to tackle the problem of semantic space evaluation and selection.

Semantic Representation and Analysis (SRA) Framework Considering common properties of the popular vector-based semantic spaces such as LSA or LDA, a formal definition of vector-based semantic spaces was introduced in order to build a unified mathematical framework to measure differences between semantic spaces (Definition 1; Hu et al., 2005): A vector-based semantic space contains five components:

A set of words X0 = { x1, . . . ,xn }. A hierarchy of layers: X1 . . . . , XM, where an element in set Xi is a finite ordered array of elements in Xi–1, i = 1, . . . , M. Vector representation for elements in each of the layers. A measure of similarity between elements within each of the layers. Maps from vector representation of layer Xi–1 to vector representation of Xi, i = 1, . . . , M. This definition can be summarized in three basic assumptions of vectorbased semantic spaces (Hu et al., 2005, 2014): Hierarchical. Semantics of different levels of a language entity may be represented differently. Algebraic. The semantics of any level of language entities must be capable of being represented numerically or algebraically. Computational. The semantic representations of a higher-level language entity are computed as a function of semantic representations for its lower-level language entities. At the lowest level of language entities, a numerical semantic comparison measure must exist between any two items (e.g., words).

With these assumptions, especially, the computational assumption, it is possible to consider any vector-based semantic space as induced semantic structure (ISS; Hu et al., 2005, 2014). When considering the ISS of a vector-based semantic space, a word is represented as its (numerical) semantic relation with other words in the space. For example, the semantic meaning of life can be expressed as nearest neighbors (Table 1 of Hu et al., 2005). Intuitively, ISS can be understood as "nearest semantic neighbors" but with several useful properties. Among them are three additional types of semantic similarity measures (combinatorial, permutational, and quantitative) that can be used to measure semantic similarity between semantic spaces (Hu et al., 2005).

Although the original goal of introducing SRA was for the purpose of evaluating the quality of semantic spaces and selecting appropriate semantic spaces from among multiple spaces produced from a different set of parameters, the ISS of SRA and the derived similarity measures made it possible for an intuitive interpretation of similarity (Hu et al., 2005) and computer implementation (Hu et al., 2014).

CONVERSATION-BASED INTELLIGENT TUTORING SYSTEMS (CBITS)

The Institute of Electrical and Electronics Engineers (IEEE) recently approved a standard committee for Adaptive Instructional Systems (AIS; P2247.1—Standard for the Classification of Adaptive Instructional Systems). This is one of the significant milestones to advance personalized learning, which is identified by the National Academy of Engineering as one of the grand challenges of the 21st century (http://www.engineeringchallenges.org/9127.aspx). Conversation-based intelligent tutoring systems (CbITS) is a class of AIS that are among the most studied and efficiently implemented in the last 20 years. One of the reasons for the advances is largely due to the use of NLP, especially the use of semantic encoding and decoding algorithms such as LSA (Graesser, Penumatsa, Ventura, Cai, & Hu, 2007).

AutoTutor as an Example Implementation of CbITS To demonstrate the use of SRA in CbITS, this chapter will use a successful implementation of CbITS called AutoTutor (Graesser et al., 2004; Nye, Graesser, & Hu, 2014; Nye, Graesser, Hu, & Cai, 2014; Person et al., 2000). AutoTutor holds conversations with humans in natural language. The authors of this chapter are among those who have developed multiple versions of AutoTutor that teach critical thinking (Graesser et al., 2010; Wallace et al., 2009), computer literacy (Graesser et al., 2004; Person, 2003), physics (Graesser et al., 2003), reading (Graesser et al., 2016), and electronics (Morgan et al., 2018).

AutoTutor applications are built with the guidance of human learning principles (Graesser, Halpern, & Hakel, 2008), such as deep questioning, to help students learn by holding deep reasoning conversations (Graesser & Person, 1994). AutoTutor converses with learners following the expectation-misconception tailored (EMT) dialogue (Graesser et al., 2004) pattern. An AutoTutor conversation often starts with a main question about a certain topic. The goal of the conversation is to help students construct an acceptable answer (expected answers) to the main question. Instead of telling students the answers, AutoTutor asks a sequence of questions (hints, prompts) that target specific concepts involved in the ideal answer to the main question. AutoTutor systems respond to students' natural language input, as well as other interactions, such as making a choice, arranging some objects in the learning environment, etc.

A Formal Framework of CbITS In this section, we introduce a simple and formal (mathematical) framework of CbITS. Again, we use AutoTutor as a typical CbITS. From the perspective of this framework, we analyze AutoTutor and then show the role of SRA in AutoTutor. The formal model is based on a behavioristic account of AutoTutor. In a typical tutoring session, AutoTutor interacts with students with the following five steps: AutoTutor (1) provides a seed question, (2) gets a response from the learner, (3) evaluates the response, (4) selects the next item, and (5) delivers the item to the learner. The minimum content requirements for a functional AutoTutor implementation would be a "bag" of items. These items are in the form of questions, pumps, prompts, hints, elaborations, or summaries (Nye, Graesser, & Hu, 2014). For each item, there may be associated elements. For example, in AutoTutor, prompt completions are elements that are used to evaluate the student's responses to the prompts. In this model, items are generically defined, such that each item can have possible purposes (diagnostic, grounding, etc.) or be a question, simple feedback, elaboration, and so on. Items are also organized such that the organization can reflect the domain knowledge and teaching knowledge of the tutor. AutoTutor always starts with a main question. The interactive dialogue between AutoTutor and Learner can be simplified into a repetition of four turns:

1. AutoTutor selects an item and presents to the student.
2. The student responds to the selected item.
3. AutoTutor evaluates the student's response.
4. AutoTutor selects the next item based on the evaluation.

Although AutoTutor always starts with a seed question, the student's response (step 3) can be a follow-up question (considered as student initiative), so AutoTutor is a typical mixed-initiative dialogue system with natural language interaction (Graesser et al., 2007). The interaction repeats until the AutoTutor select an item that does not require a response from the student. In the above simplification of tutor-student interaction, the knowledge of the tutor is represented by certain items and the relationship among them. The teaching tactics are also represented by certain items, their relations, and most importantly, the way the tutor selects them. In this model, the student model is indirectly represented by the evaluation of a stimulus-response pair. AutoTutor evaluates student's input as a semantic representation vector and its relations with associated elements (such as answers corresponding to the given items such as hints or prompts). The mathematical model that goes with the above description can be explicitly formulated as the following components.

Knowledge representation. A set X = { x1, . . . ,xn} is a set of items with an algebraic structure modeled as a subset of X × X. An item may be a collection of structured elements. Evaluation method. The evaluation method uses a function that assigns a vector of numerical values to any combination of (xi, ri), where ri is a generic notation for response of item xi, The evaluation model is generically denoted as En(i) = (xi, fn(xi, ri)), AutoTutor's evaluation of student's response ri to items xi presented on nth turn. Selection mechanism. A selection mechanism is a conditional probability distribution for any item x as the nth iteration between the tutor and student. Pn{xi} = Pr{xi|En–1(in–1), . . . ,E1(i1)}. Delivery model. A delivery model specifies how any selected item xi is presented in a given turn n. Several factors go into this model: the structure of the elements contained in the item, the frequency of how other items have been selected, and how items were presented in previous turns. The above is a general model that may be used to describe both the outer loop and the inner loop of ITS (VanLehn, 2006). The evaluation method may be different for different ITS implementations. For CbITS such as AutoTutor, the evaluation method uses similarity measures of a domain-specific vector-based semantic model.

SRA IN CBITS

We now illustrate roles that SRA could play in CbITS and critical issues that occur when SRA is used in CbITS. We continue to use AutoTutor as an illustrative example. The basic interaction mechanism of AutoTutor is guided by EMT. EMT contains a well-organized (as knowledge representation) set of items such as expectations, pumps, prompts, hints, and elaborations. Some of the items, such as expectations, prompts, and hints, require answers so they can be used to evaluate students' input. These answers in CbITS are in the form of natural language; hence SRA can serve as their semantic answers. With the formal model of CbITS outlined in the previous section, it is obvious that SRA can be used in the first two of the four components, knowledge representation and evaluation method. For example, an expectation for an EMT in ElectronixTutor (Graesser, Hu, et al., 2017) could be: In cut-off mode, the transistor works as an open switch when both the emitter and collector junctions are reverse-biased.

Some expected answers may be typical good answers, and some of them are typical bad answers:

Good Answer 1: In cut-off mode, the transistor works as an open switch when both emitter and collector junctions are reverse-biased.

Bad Answer 1: In cut-off mode, the transistor works as a closed switch when both emitter and collector junctions are forward-biased.

Bad Answer 2: In cut-off mode, the transistor works as an amplifier.

Bad Answer 3: In cut-off mode, both emitter and base junctions are reverse-biased.

For each of the expectations, there are multiple hints and prompts. For example, this is a hint that helps students understand the expected answers:

Under what junction bias conditions will the transistor operate as an open switch in the cut-off mode? Corresponding to this hint, there are possible good answers and bad answers.

Good Answer: When both emitter and collector junctions are reverse-biased.

Bad Answer 1: When both emitter and collector junctions are forward-biased.

Bad Answer 2: When both emitter and base junctions are reverse-biased.

In the same way, prompts are similar to hints, but the answers are shorter. For example, this is a prompt: Both emitter and collector junctions are biased in what direction in cutoff mode, when the transistor works as an open switch? With very short answers:

Good Answer: In reverse-biased

Bad Answer: In forward-biased

In AutoTutor EMT dialogue (Graesser et al., 2004), multiple prompts and hints are associated with each expectation. It is obvious that SRA can be used to represent semantic keys for all the answers. The semantic keys on one hand are used to create relationship between these items, so SRA may help to build knowledge representation. Most importantly, it is used to compare answers with students' natural language contributions.

Requirements for SRA in CbITS From the above example, we observe that (1) answers to expectations, hints, and prompts are in the form of natural language, and (2) good answers and bad answers only differ in keywords. For example, the expected answers for the hint “Under what junction bias conditions will the transistor operate as an open switch in the cut-off mode?” are “When both emitter and collector junctions are reverse-biased” (good answer) and “When both emitter and collector junctions are forward-biased” (bad answer). For most of the widely used semantic encoding and decoding methods, it is not easy to distinguish highly similar texts when large portions of the two texts are the same (Hu et al., 2007). In addition, efficient ITS applications are domain-specific, interaction between ITS and students should be context-sensitive, and most desirable, adaptive and individualized. With these requirements for an ideal CbITS, effective and efficient SRA should have additional attributes: SRA should be domain specific. The examples of SRA we have described in this chapter are generic framework for semantic processing. Most of the widely used semantic encoding and decoding methods share the common properties. They are started with a large body of texts. The selection of the texts is the key for domain specificity. For example, a semantic representation for term plane in mathematics (Figure 4.1a) should be different from that in a general domain (Figure 4.1b). The key steps for domain-specific semantic encoding are the first two steps, namely selection of a domain and selection of corpora.

SRA should be used in conjunction with other methods. Very often, typical good answers and bad answers only differ in key terms or presence or absence of negations. For example, "When both emitter and collector junctions are reverse-biased." and "When both emitter and collector junctions are forward-biased." are a good answer and a bad answer for a hint. SRA's evaluation of students' input will not be sensitive because the two answers have high levels of similarity (common texts; Hu et al., 2007). It would be easier and more efficient to use other methods such as regular expression (Thompson, 1968) for comparison.

DISCUSSION

Enabling computers to understand natural language and converse with humans has been a continuous effort for generations of scientists. From the computer therapist Eliza (Weizenbaum et al., 1966) 50 years ago to the most recent implementation of AutoTutor (Graesser et al., 2004; Nye, Graesser, & Hu, 2014), the key enabling technology is to let computers efficiently compare similarity between texts.

FIGURE 4.1 ISS for plane (a) created from a large corpus of mathematics-related texts and generated using the seeding methods (Cai et al., 2018), and (b) semantic space created from a general purpose text corpus .

There are different ways to compute similarity between texts. The easiest and most intuitive way is to count the number of common words. The more sophisticated way is to semantically encode terms, sentences, and documents for each of the texts and then compute their semantic similarity. To apply the semantic encoding technology in learning science, there are two challenges: (1) how fast to generate a domain-specific and contextsensitive semantic representation for a given application, and (2) how to systematically evaluate quality of the semantic representation for a given application. This chapter tries to offer a solution for both challenges by introducing a general framework of semantic representation. High quality domain-specific and context sensitive semantic representation enable an effective and efficient conversation-based intelligent tutoring system (CbITS).

CbITS plays a very important role in learning science and technology. CbITS implementations are directly inspired by human tutors. The current CbITS prototypes such as AutoTutor (Graesser et al., 2004; Nye, Graesser, & Hu, 2014) are built with the guide of learning principles from cognitive psychology. They are proven effective and behaviorally similar to human tutors (VanLehn et al., 2007). In addition to CbITS, the SRA framework may also be used in other applications and technologies in learning science. For example, it can serve as general assessment framework to assess learners' knowledge and build context-specific student models (Hu, Morrison, & Cai, 2013) or to build student models in a team tutoring environment (Hu et al., 2018).

CONCLUSIONS

Semantic encoding and decoding is one of the essential enabling technologies for conversation-based intelligent tutoring system (CbITS). There are various ways to computationally represent the semantics of a piece of text. In this chapter we used a general framework called Semantic Representation and Analysis (SRA). The definition of SRA is general enough to capture most of the widely used semantic encoding and decoding methods used in text mining. For illustrative purposes, we used latent semantic analysis (LSA) as a typical instance of SRA to describe the process of creating domain-specific semantic representation. In this chapter, we used AutoTutor as an typical example of CbITS. AutoTutor is an intelligent tutoring system that holds conversations with humans in natural language. To explain how would SRA be used in

CbITS, we used an alternative model of ITS and argue that SRA is most useful in knowledge representation of the domain and in enhancement of evaluation methods in CbITS. Because CbITS applications are mostly domain specific and conversational in nature (short answers with similar good or bad answers), we suggest that SRA for any CbITS be domain specific and used in conjunction with other methods.

REFERENCES

Aleven, V. A. W. M. M., & Koedinger, K. R. (2002). An effective metacognitive strategy: learning by doing and explaining with a computer-based Cognitive Tutor. Cognitive Science, 26(2), 147–179.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. Journal of the Learning Sciences, 4(2), 167–207.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003, January). Latent Dirichlet Allocation. Jour nal of Machine Learning Research: JMLR, 3, 993–1022.

Burgess, C. (1998). From simple associations to the building blocks of language: Modeling meaning in memory with the HAL model. Behavior Research Methods, Instruments, & Computers: A Journal of the Psychonomic Society, Inc, 30(2), 188–198.

Cai, Z., Cheng, Q., Graesser, A. C., Shaffer, D. W., Windsor, L. C., & Hu, X. (2018). Impact of corpus size and dimensionality of LSA spaces from Wikipedia articles on AutoTutor answer evaluation. In K. E. Boyer & M. Yudelson (Eds.), educational data mining (pp. 127–136). Buffalo, NY: EDM Society.

Cai, Z., Li, H., Hu, X., & Graesser, A. (2016). Can word probabilities from LDA be simply added up to represent documents? In EDM (pp. 577–578). educationaldatamining.org

Cai, Z., McNamara, D. S., Louwerse, M., Hu, X., Rowe, M., Graesser, A. C., . . . Others. (2004). NLS: A non-latent similarity algorithm. In Proceedings of the Annual Meeting of the Cognitive Science Society (Vol. 26). cloudfront.escholarship.org. Retrieved from https://cloudfront.escholarship.org/dist/prd/content/qt0sc5p977/qt0sc5p977.pdf

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6), 391–407.

du Boulay, B., & Luckin, R. (2001). Modelling human teaching tactics and strategies for tutoring systems. International Journal of Artificial Intelligence in Education, 12(3), 235–256.

Dumais, S. T., Furnas, G. W., Landauer, T. K., Deerwester, S., & Harshman, R. (1988). Using latent semantic analysis to improve access to textual information. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 281–285). New York: ACM.

Foltz, P. W., Laham, D., & Landauer, T. K. (1999). Automated essay scoring: Applications to educational technology. In EdMedia: World Conference on Educational Media and Technology (pp. 939–944). Association for the Advancement of Computing in Education (AACE).

Franceschetti, D. R., Karnavat, A., Marineau, J., McCallie, G. L., Olde, B. A., Terry, B. L., & Graesser, A. C. (2001). Development of physics text corpora for latent semantic analysis. In Proceedings of the 23rd annual conference of the cognitive science society (pp. 297–300). Erlbaum Mahwah, NJ.

Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., & Lochbaum, K. E. (1988). Information retrieval using a singular value decomposition model of latent semantic structure. In Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 465–480). New York: ACM.

Gabrilovich, E., & Markovitch, S. (2007). Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In IJcAI 7, 1606–1611). aaai.org.

Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negativesampling word-embedding method. arXiv [cs.CL]. Retrieved from http://arxiv.org /abs/1402.3722

Golub, G., & Kahan, W. (1965). Calculating the singular values and pseudo-inverse of a matrix. Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis, 2(2), 205–224.

Graesser, A., Britt, A., Millis, K., Wallace, P., Halpern, D., Cai, Z., . . . Forsyth, C. (2010). Critiquing media reports with flawed scientific findings: Operation ARIES! A Game with animated agents and natural language trialogues. In intelligent tutoring systems (pp. 327–329). Berlin, Heidelberg: Springer.

Graesser, A. C., Cai, Z., Baer, W. O., Olney, A. M., Hu, X., Reed, M., & Greenberg, D. (2016). Reading comprehension lessons in AutoTutor for the Center for the Study of Adult Literacy. Adaptive educational technologies for literacy instruction, 288–293.

Graesser, A. C., Dowell, N., & Clewley, D. (2017). Assessing collaborative problem solving through conversational agents. In A. A. Davier, M. Zhu, & P. C. Kyllonen (Eds.), Innovative assessment of collaboration (pp. 65–80). Springer International Publishing.

Graesser, A. C., Halpern, D. F., & Hakel, M. (2008). 25 principles of learning. Washing ton, DC: Task Force on Lifelong Learning at Work and at Home.

Graesser, A. C., Hu, X., Nye, B. D., VanLehn, K., Kumar, R., Heffernan, C., . . . Others. (2017). ElectronixTutor: an intelligent tutoring system with multiple learning resources for electronics. International Journal of STEM Education: Innovations and Research.

Graesser, A. C., Hu, X., & Person, N. (2001). Teaching with the help of talking heads. In Proceedings IEEE International Conference on Advanced Learning Technologies (pp. 460–461).

Graesser, A. C., Jackson, G. T., Matthews, E. C., Mitchell, H. H., Olney, A., Ventura, M., . . . Others. (2003). Why/AutoTutor: A test of learning gains from a physics tutor with natural language dialog. In Proceedings of the Annual Meeting of the Cognitive Science Society (Vol. 25). cloudfront.escholarship.org. Retrieved from https:// cloudfront.escholarship.org/dist/prd/content/qt6mj3q2v1/qt6mj3q2v1.pdf

Graesser, A. C., Lu, S., Jackson, G. T., Mitchell, H. H., Ventura, M., Olney, A., & Louwerse, M. M. (2004). AutoTutor: a tutor with dialogue in natural language. Behavior Research Methods, Instruments, & Computers: A Journal of the Psychonomic Society, Inc, 36(2), 180–192.

Graesser, A. C., Penumatsa, P., Ventura, M., Cai, Z., & Hu, X. (2007). Using LSA in AutoTutor: Learning through mixed initiative dialogue in natural language. Handbook of Latent Semantic Analysis, 243–262.

Graesser, A. C., & Person, N. K. (1994). Question asking during tutoring. American Edu cational Research Journal, 31(1), 104. Harris, Z. S. (1954). Distributional structure. Word & World, 10(2–3), 146–162.

Hoffman, M., Bach, F. R., & Blei, D. M. (2010). Online learning for latent Dirichlet allocation. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, & A. Culotta (Eds.), Advances in Neural Information Processing Systems 23 (pp. 856–864). Curran Associates, Inc.

Hu, X., Cai, Z., Franceschetti, D., Penumatsa, P., Graesser, A. C., Louwerse, M. M., & McNamara, D. S. (2003). LSA: First dimension and dimensional weighting. In Proceedings of the Annual Meeting of the Cognitive Science Society (Vol. 25). cloudfront.escholarship.org. Retrieved from https://cloudfront.escholarship.org/dist /prd/content/qt0p3526z0/qt0p3526z0.pdf

Hu, X., Cai, Z., Graesser, A. C., & Ventura, M. (2005). Similarity between semantic spaces. In Proceedings Of The 27th Annual Conference Of The Cognitive Science Society (pp. 995–1000). Hillsdale, NJ: LEA.

Hu, X., Cai, Z., Louwerse, M., Olney, A., Penumatsa, P., & Graesser, A. (2003). A revised algorithm for latent semantic analysis. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (pp. 1489–1491). San Francisco, CA: Morgan Kaufmann Publishers Inc.

Hu, X., Cai, Z., Wiemer-Hastings, P., Graesser, A. C., & McNamara, D. S. (2007). Strengths, limitations, and extensions of LSA. The Handbook of Latent Semantic Analysis, 401–426.

Hu, X., Dowell, N., Cai, Z., Graesser, A. C., Shi, G., Cockroft, J. L., & Shorter, P. (2018). Constructing individual conversation characteristics curves (ICCC) for interactive intelligent tutoring environments (IITE). In R. Sottilare, A. Graesser, X. Hu, and A. M. Sinatra (Ed.), Design recommendations for intelligent tutoring systems: Volume 6—team tutoring (Vol. 6, p. 133). US Army Research Laboratory.

Hu, X., Morrison, D. M., & Cai, Z. (2013). Conversation-based intelligent tutoring system (Vol. 1, p. 97). US Army Research Laboratory. Hu, X., Nye, B. D., Gao, C., Huang, X., Xie, J., & Shubeck, K. (2014). Semantic representation analysis: A general framework for individualized, domain-specific and context-sensitive semantic processing. In Foundations of augmented cognition: Advancing human performance and decision-making through adaptive systems (pp. 35–46). Springer, Cham.

Landauer, T. K. (2003). Automatic essay assessment. Assessment in Education: Principles, Policy & Practice, 10(3), 295–308. Landauer, T. K. (2006). Latent semantic analysis. In Encyclopedia of cognitive science. John Wiley & Sons, Ltd.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. Discourse Processes, 25(2–3), 259–284. Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (2013). Handbook of latent semantic analysis. Psychology Press.

Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. MIT Press.

Morgan, B., Hampton, A. J., Cai, Z., Tackett, A., Wang, L., Hu, X., & Graesser, A. C. (2018). ElectronixTutor integrates multiple learning resources to teach electronics on the web. In Proceedings of the Fifth Annual ACM Conference on Learning at Scale (pp. 33:1–33:2). New York: ACM.

Nelson, D. L., McEvoy, C. L., & Schreiber, T. A. (2004). The University of South Florida free association, rhyme, and word fragment norms. Behavior Research Methods, Instruments, & Computers: A Journal of the Psychonomic Society, Inc, 36(3), 402–407.

Nye, B. D., Graesser, A. C., & Hu, X. (2014). AutoTutor and family: A review of 17 years of natural language tutoring. International Journal of Artificial Intelligence in Education, 24(4), 427–469.

Nye, B. D., Graesser, A. C., Hu, X., & Cai, Z. (2014). AutoTutor in the cloud: A serviceoriented paradigm for an interoperable natural-language ITS. Journal of Advanced Distributed Learning Technology, 2(6), 35–48.

Olde, B. A., Franceschetti, D. R., Graesser, A. C., & Karnavat, A. (2002). The right stuff: Do you need to sanitize your corpus when using latent semantic analysis? In Proceedings of the Annual Meeting of the Cognitive Science Society (Vol. 24). cloudfront.escholarship.org. Retrieved from https://cloudfront.escholarship.org/dist /prd/content/qt56m7s0j2/qt56m7s0j2.pdf

Olney, A., & Cai, Z. (2005a). An orthonormal basis for entailment. In FLAIRS Confer ence (pp. 554–559). aaai.org.

Olney, A., & Cai, Z. (2005b). An orthonormal basis for topic segmentation in tutorial dialogue. In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing: Association for Computational Linguistics.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1532–1543). aclweb.org.

Person, N. K. (2003). AutoTutor improves deep learning of computer literacy: Is it the dialog or the talking head? Artificial Intelligence in Education: Shaping the Future of Learning Through Intelligent Technologies, 97, 47.

Person, N. K., Craig, C., Price, P., Hu, X., Gholson, B., & Graesser, A. C. (2000). Incorporating human-like conversational behaviors into AutoTutor. In Proceedings of the Workshop on Achieving Human-like Behavior in the Interactive Animated Agents at the Agents 2000 Conference (pp. 85–92).

Riordan, B., & Jones, M. N. (2011). Redundancy in perceptual and linguistic experience: Comparing feature-based and distributional models of semantic representation. Topics in Cognitive Science, 3(2), 303–345. Rong, X. (2014). word2vec Parameter Learning Explained. arXiv [cs.CL]. Retrieved from http://arxiv.org/abs/1411.2738

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613–620.

Shams, M., & Baraani-Dastjerdi, A. (2017). Enriched LDA (ELDA): Combination of latent Dirichlet allocation with word co-occurrence analysis for aspect extraction. Expert Systems with Applications, 80, 136–146.

Thompson, K. (1968). Programming Techniques: Regular Expression Search Algorithm. Communications of the ACM, 11(6), 419–422.

VanLehn, K. (2006). The behavior of tutoring systems. International Journal of Artificial Intelligence in Education, 16(3), 227–265.

VanLehn, K., Graesser, A. C., Jackson, G. T., Jordan, P., Olney, A., & Rosé, C. P. (2007). When are tutorial dialogues more effective than reading? Cognitive Science, 31(1), 3–62.

VanLehn, K., Jones, R. M., & Chi, M. T. H. (1992). A model of the self-explanation effect. Journal of the Learning Sciences, 2(1), 1–59.

Wallace, P. S., Graesser, A. C., Millis, K. K., Halpern, D. F., Cai, Z., Britt, M. A., . . . Wiemer, K. (2009). Operation ARIES! A computerized game for teaching scientific inquiry. In AIED (pp. 602–604). niu.edu.

Weizenbaum, J., et al. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36–45.

Wolfe, M. B. W., Schreiner, M. E., Rehder, B., Laham, D., Foltz, P. W., Kintsch, W., & Landauer, T. K. (1998). Learning from text: Matching readers and texts by latent semantic analysis. Discourse Processes, 25(2–3), 309–336.

Yu, M., & Dredze, M. (2014). Improving lexical embeddings with semantic knowledge. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 545–550). aclweb.org.