UNIVERSIDADE DE SÃO PAULO

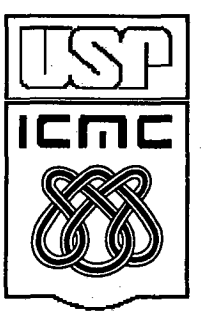
Instituto de Ciências Matemáticas e de Computação

A Statistical Generative Model for Unsupervised Learning of Verb Argument Structures

Thiago Alexandre Salgueiro Pardo
Maria das Graças Volpe Nunes

Nº 84

NOTAS



São Carlos - SP

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2577

A Statistical Generative Model for Unsupervised Learning of Verb Argument Structures

Thiago Alexandre Salgueiro Pardo
Maria das Graças Volpe Nunes

Nº 84

NOTAS

Série Computação

São Carlos – SP
Fev./2005

Universidade de São Paulo - USP
Universidade Federal de São Carlos - UFSCar
Universidade Estadual Paulista - UNESP

A Statistical Generative Model for Unsupervised Learning of Verb Argument Structures

Thiago Alexandre Salgueiro Pardo
Maria das Graças Volpe Nunes

NILC-TR-05-03

Fevereiro, 2005

Série de Relatórios do Núcleo Interinstitucional de Lingüística Computacional
NILC - ICMC-USP, Caixa Postal 668, 13560-970 São Carlos, SP, Brasil

A Statistical Generative Model for Unsupervised Learning of Verb Argument Structures

Thiago Alexandre Salgueiro Pardo
Maria das Graças Volpe Nunes

Núcleo Interinstitucional de Lingüística Computacional (NILC)
CP 668 – ICMC-USP, 13.560-970 São Carlos, SP, Brasil
http://www.nilc.icmc.usp.br
{thiago@nilc.icmc.usp.br; gracan@icmc.usp.br}

Abstract. We present a statistical generative model for unsupervised learning of verb argument structures. We use the model in order to automatically induce verb argument structures for a representative set of verbs. Approximately 80% of the induced argument structures are judged correct by human subjects. The structures overlap significantly with those in PropBank; they also exhibit correct patterns of usage that are not present in this manually developed semantic resource.

Resumo. *Apresenta-se, neste relatório, um modelo estatístico gerativo para o aprendizado não supervisionado das estruturas argumentais dos verbos. Este modelo é usado para induzir as estruturas argumentais de um conjunto representativo de verbos. Aproximadamente 80% destas estruturas induzidas foram julgados corretos por avaliadores humanos. Estas estruturas correspondem significativamente às estruturas previstas pelo PropBank, apresentando, também, padrões de uso corretos que não estão presentes neste repositório semântico desenvolvido manualmente.*

1 Introduction

Inspired by the impact that the availability of the Penn Treebank (Marcus et al., 1993; Marcus, 1994) had on syntactic parsing, several efforts have recently focused on the creation of semantically annotated resources. The annotation of verb arguments, their roles, and preferential linguistic behaviors (Levin, 1993) represents a significant fraction of these efforts.

The annotations that we are focusing on here pertain to the argument structures of a verb. These annotations describe canonical conceptual structures that underlie verb usage, indicating the number of arguments of the verb, in which order the arguments appear in a sentence and which types of argument are possible. For instance, for the verb "buy", a plausible structure is buy(person,thing), indicating that (a) the verb "buy" asks for 2 arguments, (b) the first argument is of type person, and (c) the second argument is of type thing. This structure underlies, for example, the sentence "John bought a car".

Little agreement exists with respect to how many canonical usages a verb has, what their required arguments are, and what argument orderings are employed more often in real texts. Examples (1)-(3) show some patterns of usage for the verb (or predicate) "bought".

(1) He had bought them gifts.
(2) He bought it 40 years ago.
(3) About 8 million home water heaters are bought each ear.

In these examples, note that: the object/thing that is bought ("gifts" in sentence (1), "it" in sentence (2), and "about 8 million home water heaters" in sentence (3)) is more likely to be a required argument for the verb than the time when the buying event occurred; in (1) and (2), the thing bought is stated after the verb, while the opposite happens in (3). Ideally, all these possibilities should be acknowledged in the semantic specification of verbs.

There is also little agreement about how the arguments should be labeled. Figures 1, 2, and 3 show the information associated with the verb "buy" in FrameNet (Baker et al., 1998), VerbNet (Kipper et al., 2000), and PropBank (Kingsbury and Palmer, 2002), respectively. These are large scale projects that aim at developing semantic information repositories for verbs, mainly.

FrameNet shows the pattern in which a verb occurs and provides representative examples; the resource also organizes the verbs in a hierarchy that implicitly encodes how verb arguments can be inherited from ancestors. VerbNet shows the thematic roles the verb asks for, their semantic features, and possible subcategorization frames[1]; VerbNet also provides examples for each categorization frame. PropBank makes explicit the argument roles of a verb, the possible subcategorization frames, and provides examples for each

[1] It is important to make clear a basic distinction between argument structures and subcategorization frames: while the argument structures of a verb indicate its underlying semantic behavior, the subcategorization frames indicate the possible syntactic contexts that the verb imposes in the sentences in which it occur. There are several proposals of what knowledge these two levels should encode and how they should interact and be represented. For the purposes of this work, it suffices to assume the simple distinction we made before.

one. PropBank also distinguishes between obligatory verb complements (the arguments) and optional ones (adjuncts). The optional complement in Figure 3, for example, is the ArgM-MNR argument (i.e., argument of manner).

By inspecting Figures 1-3, it is possible to see the differences between the ontological status of argument labels. What is ARG1 after all? "Goods"? A "Theme"? Or the "Thing Bought"? What is the most appropriate level of abstraction for argument labels? It is also easy to note that only PropBank marks the adjunct/argument distinction, while FrameNet and VerbNet do not.

Typical pattern:

BUYER buys GOODS from SELLER for MONEY

Example:

Abby bought a car from Robin for $5,000.

Figure 1. FrameNet annotation of the verb "buy"

Thematic Roles:
Agent[+animate OR +organization]
Asset[-location -region]
Beneficiary[+animate OR +organization]
Source[+concrete]
Theme[]

Frames:

Basic Transitive:

"Carmen bought a dress"
Agent V Theme

Benefactive Alternation (double object):

"Carmen bought Mary a dress"
Agent V Beneficiary Theme

Figure 2. VerbNet annotation of the verb "buy"

Roles:

Arg0:buyer
Arg1:thing bought
Arg2:seller
Arg3:price paid
Arg4:benefactive

Examples:

Intransitive:

Consumers who buy at this level are more educated than they were.

Arg0:	Consumers
REL:	buy
ArgM-MNR:	at this level

Basic transitive:

They bought $2.4 billion in Fannie Mae bonds

Arg0:	They
REL:	bought
Arg1:	$2.4 billion in Fannie Mae bonds

Figure 3. PropBank annotation of the verb "buy"

Given the difficulty of the task, it is not surprising that FrameNet, VerbNet, and PropBank have been manually built. However, some research efforts have targeted the problem of automatic (Brent, 1991; Resnik, 1992; Grishman and Sterling, 1992; Manning, 1993; Framis, 1994; Briscoe and Carroll, 1997) and semi-automatic (Green et al., 2004; Gomez, 2004) verb structure induction. In general, these approaches either rely on syntactic parsers and/or subcategorization dictionaries for identifying the arguments of a verb in a sentence, or assume as known the structure types (in terms of number and order of arguments a verb can assume). The main goal in these approaches is to identify the lexemes that are most likely to fill a given verb argument slot. Some researchers (Grishman and Sterling, 1994; Framis, 1994; Gomez, 2004) try to go beyond these lexemes and generalize the structures that are learned, by computing the similarity between the words occurring across similar structure instances or by using lexical resources such as WordNet[2] (Fellbaum et al., 1999). Most of these approaches implement a filtering step, in which inadequate learned structures are discarded manually or automatically on frequency-based grounds.

[2] www.cogsci.princeton.edu/~wn/

More recent approaches (Rooth et al., 1999; McCarthy, 2000; Gildea, 2002) are based on probabilistic models that cluster similar verbs according to the arguments they ask for (see (Levin 1993) for diathesis alternation studies) and that associate classes to arguments in the structures, in order to make generalizations. These models also depend heavily on the availability of syntactic parsers.

The drawbacks of the previous approaches are clear:

- They often assume that the argument structure types that characterize a given verb are known in advance.
- They make use of sophisticated resources, such as syntactic parsers, WordNets, and subcategorization dictionaries. For some languages and domains, such resources are either unavailable or insufficiently accurate.

In this paper, we propose an unsupervised approach to verb argument structure induction, which has the following advantages over previous work:

- Our approach does not assume that the canonical verb argument structures are known in advance. Rather, the canonical structures are automatically learned from naturally occurring texts, concurrently with the induction of the most likely lexemes and abstractions (named entities) that act as argument slot fillers.
- In addition to learning the most likely verb argument structures, our approach also ranks competing structures according to their probability. In our approach, not only do we know that two canonical argument structures are likely to be used in conjunction with a given verb, but we also know that one structure is 50 times more likely, for example, than the other.
- Our approach makes use of simple tools, such as part of speech and named entity taggers, that are both widely available and easy to port across languages and domains.
- The argument structures that we learn are grounded in both lexemes and abstractions – named entities – that are easy to operate within the context of many other NLP applications.

We developed a statistical model based on Shannon's noisy-channel model (1948), which is trained with the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). By model, we mean a theory/construct that try to explain how real world events/processes happen. By statistical (or probabilistic) model, we mean a model that uses probabilities to determine how and which events/processes happen (Manning and Schütze, 1999).

In the rest of the paper, we first introduce the noisy-channel model (Section 2) and the EM algorithm (Section 3). Then, we describe our statistical model (Section 4) and the data used for training it (Section 5). A

human-based evaluation of the verb argument structures that we learn automatically is presented (Section 6). We end with a discussion of the strengths and weaknesses of our model and future work (Section 7).

2 The noisy-channel model

The noisy-channel model was proposed by Shannon (1948) for modeling the transmission capacity of data in a channel. Originally, this model was used in telephone systems for trying to predict and correct errors that eventually occurred during the messages transmission. Figure 4 shows the model components. Initially, there is a message M1 produced by a source with probability P(M1); this message is transmitted in a noisy channel and is corrupted, becoming a message M2 with probability P(M2|M1).

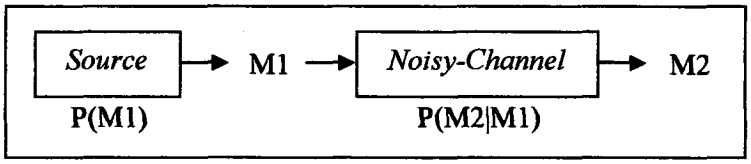


Figure 4. Noisy-channel model

In telephone systems, the source is usually a person and the noisy channel is the telephone line. According to this model, if probabilities P(M1) and P(M2|M1) are known, it is possible to determine M1 from M2 by means of a decoding process. This process consists in choosing the message M1 that maximizes the probabilities P(M1) and P(M2|M1).

The noisy-channel model was applied to the speech processing area with success, and, recently, it has been applied to written language processing tasks. Specially, this model advanced the state of the art in statistical machine translation, producing the best translators in the field, according to the international evaluations carried out by NIST[3] (National Institute of Standards and Technology). In this task, the translation of a French sentence to English, for instance, is modeled in the following way: a English sentence E is produced by a person (source) with probability P(E), and, during its communication (through a hypothetical noisy channel), it becomes a French sentence F with probability P(F|E). The translation of F into E consists, therefore, in a decoding process, considering that P(E) and P(F|E) are known or can be learned.

In Natural Language Processing (NLP), the set of probabilities P(E) is called linguistic model, because it tells the probability of E occurs in a

[3] http://www.nist.gov/

language; the set of probabilities P(F|E) is called translation model, because it indicates how E becomes/is translated into F. This translation process of E into F is the main part of the noisy-channel model and is called generative story or model. In general, the success of a noisy-channel model in a task depends on the generative story that is assumed. For illustration of a generative story, consider the story usually assumed by simple machine translation systems from English to Portuguese to the sentence "Mary did not slap the green witch":

(1) Initially, some words are eliminated or replicated a specific number of times (the word "did" was eliminated, while the word "slap" was replicated twice)

Mary not slap slap slap the green witch

(2) Then, each English word is substituted by a Portuguese word (in sequence, "Mary" by *Maria*, "not" by *não*, first occurrence of "slap" by *deu*, second occurrence of "slap" by *um*, third occurrence of "slap" by *tapa*, "the" by *na*, "green" by *verde*, and "witch" by *bruxa*)

Maria não deu um tapa na verde bruxa

(3) Finally, the words are ordered, producing the sentence in Portuguese

Maria não deu um tapa na bruxa verde

The decisions of eliminating or replicating a word, substituting English words by their corresponding Portuguese words, and how to order the words in the sentence are based on the translation model probabilities, i.e., the set of probabilities P(F|E). These probabilities are called parameters and, in general, are learned from a parallel corpus by using the EM algorithm, which is introduced bellow.

For more details about the noisy-channel model, it is suggested to the reader the reference work or Manning and Schütze (1999).

3 The EM algorithm

The EM (Expectation-Maximization) algorithm (Dempster et al., 1977) is generally used to estimate the parameters of statistical models in which there are unobserved variables. If all the variables/data of the problem were observed, it would be simple to estimate the parameters of the underlying

model (by frequency counts or other machine learning technique). When some variable/data is not available, EM is applied.

In statistical machine translation, as shown before, it is assumed that an English sentence translates into a Portuguese sentence if and only if it is possible to align their words, that is, to determine the words in the Portuguese sentence that correspond to the words in the English sentence. In this case, the alignment between the sentences is an unobserved variable. Note that there are several possible alignments between two sentences, since it is necessary to consider all words correspondence possibilities (for instance, in the previous example, "Mary" with *Maria*, "Mary" with *não*, "Mary" with *deu*, ... , "witch" with *Maria*, "witch" with *não*, etc.). If these words correspondence parameters were known, it would be possible to determine the best alignment between two sentences; otherwise, if the correct alignment between two sentences were known, it would be possible to estimate the parameters of the model. In cases like this one, in which none of these data is available because of an unobserved variable, EM is used.

EM works in the following way:

1. initial probabilities are assigned to all the parameters (usually, the uniform distribution is used, i.e., all the parameters have the same probability);
2. the probabilities of all the possible alignments between all English-Portuguese sentences pairs from the training corpus are determined, in which the probability of an alignment is the multiplication of all the parameters that compose the alignment (for instance, for the sentences pair from the previous section, considering only the words correspondence parameters, the probability of the considered alignment is t(*Maria*|"Mary") x t(*não*|"not") x t(*deu*|"slap") x t(*um*|"slap") x t(*tapa*|"slap") x t(*na*|"the") x t(*verde*|"green") x t(*bruxa*|"witch"), in which t is the word correspondence parameter, i.e., the word translation parameter);
3. based on the alignments probabilities, new probabilities for the parameters are estimated;
4. based on the new parameters probabilities, new alignments probabilities are calculated;
5. and so on.

The E-step (Expectation) of EM algorithm is the calculation of the alignments probabilities; the M-step (Maximization) is the parameters estimation based on the alignments probabilities. EM guarantees that, after each iteration of the algorithm, the parameters probabilities are closer to the ideal values. EM stops when probabilities converge and stop changing significantly.

The basis of EM learning consists in pattern repetition in the training corpus. Each time a pattern is repeated, more evidence is available to EM for increasing the probability of specific parameters, penalizing the parameters that represent unlikely or inadequate events. For instance, in all the sentences

in the training corpus, it is probable that there are several alignments between the words "witch" and *bruxa*; for each pattern of this that is found, the probability of the parameter t(*bruxa*|"witch") is increased, while the probability of other parameters, like t(*Maria*|"witch"), t(*deu*|"witch"), ... , and t(*verde*|"witch"), are not, being, therefore, penalized.

One of the problems of EM algorithm is that it has exponential complexity and, considering the amount of necessary data for its training (mainly in NLP), its application becomes not viable. There are several forms to deal with this problem. Most of researches decrease the number of possible values for the unobserved variables and the amount of training data. For instance, for machine translation, it is usual to consider only the most probable alignments between two sentences (instead of all of them) and to limit the size of sentences for a specified number of words.

For more details about EM, it is suggested to the reader the reference work or Manning and Schütze (1999). In the next section, we describe the statistical model we developed for verb argument structure learning.

4 Our approach

We couch our learning problem in a probabilistic noisy-channel framework (see Figure 5). In this model, we delineate how a sentence S is generated from an underlying argument structure A. The generative store is shown bellow:

1. a) The head (verb) of the sentence is first chosen with probability P(v).
   b) The number of arguments the verb takes is chosen with probability narg(no_arg|v).
   c) Each argument is generated with probability arg(argument|v). The argument can be either an abstraction/concept (named entity in our case) or a word/lexeme.
2. Once the verb argument structure is generated, a probabilistic parameter phi(N|v) decides the number of extra words/concepts that are going to be eventually produced in the sentence.
3. Each extra word/concept is stochastically generated according to the distribution ew(word).
4. If the generative process produces concepts c (named entities), these are translated into words, with probability t(word|c).
5. All words coming from both the verb argument structure and the extras are re-ordered to produce a linear, grammatical sentence.

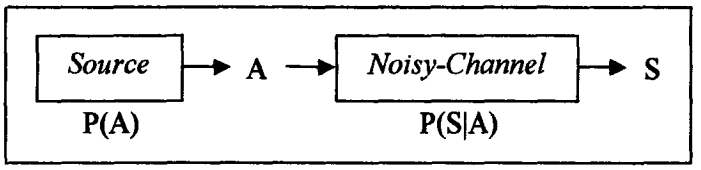


Figure 5. A noisy-channel model for learning verb argument structures

For instance, the sentence "Santa has bought them gifts" is generated by the following process:

1. The head "bought" is chosen with probability P(bought). The verb is associated with 3 arguments: PERSON1, "gifts", and PERSON2, which correspond to the following verb argument structure: bought(PERSON1, gifts, PERSON2).
2. The parameter phi adds one extra word
3. which turns out to be the word "has".
4. The named entities are translated into words: PERSON1 into "Santa" and "PERSON2" into "them".
5. All words are reordered to produce the sentence "Santa has bought them gifts".

In order to make the training of our model tractable, we make some simplifying assumptions. That is, we assume that the subsequence corresponding to steps 1.a-c happens in one shot: an entire event is generated stochastically with probability event(verb(arg_1,…,arg_n)). Since named entity taggers work at levels of accuracy above 90%, we also assume that it is not necessary to translate concepts into words as part of the generative process – we can pre-tag the sentences used for training with named entity tags and learn argument structures that include such entities directly. From a generative story perspective, this means that we no longer need Step 4 to model the translation of entities into words. As further simplifying assumption, we assume that the reordering of the words/named entities into sentences (Step 5) is governed by a uniform distribution. Mathematically, these choices simplify our model tremendously.

According to this model, the probability of a sentence S is thus given by the following formula:

$$P(S) = \sum_{A} P(S,A)$$

$$= \sum_{A} P(A) \times P(S|A)$$

$$= \sum_{A} event(A) \times phi(N|verb) \times \prod_{i=1}^{N} ew(w_i)$$

where A is a possible argument structure, N is the number of extra words/concepts that are generated, and w_i is the i^{th} extra word being generated. In this view, the probability of the sentence P(Santa/PERSON1 has bought them/PERSON2 gifts) is event(bought(PERSON1,gifts,PERSON2) × phi(1|bought) × ew(has).

We use the Expectation-Maximization (EM) algorithm to estimate the parameters of the model. In our model, the unobserved variable is the argument structure of a sentence, i.e., which are the arguments of the verb in the sentence. To restrict the search space, we assume that a verb can have at most 3 arguments and that arguments can be only open class words (adjectives, verbs, adverbs, and nouns – including pronouns). For identifying these classes, we pre-tag the data with a part of speech tagger (Ratnaparki, 1996), which is one of the most accurate taggers for English. With these restrictions into place, we are capable of doing full EM training on our data, as the number of hidden argument structures that we have to consider for every sentence is reasonable. For example, Figure 6 shows all possible hidden argument structures for the sentence "He has bought them gifts". The arrows leave from the verb and point to the arguments. The words not pointed to by any arrow are the extra words. For simplicity, the named entity and part of speech tags are not shown.

Because we use EM, low probabilities are naturally assigned to uncommon or, hopefully, inadequate argument structures. Therefore, it is not necessary to filter our results in an ad-hoc manner.

It is interesting to notice that, after the model is trained and the parameters are learned, the parameter event is the only one that encodes the information we look for, that is, the possible verb argument structures. Although we do not need the information conveyed by the parameters phi and ew, it does not mean that they are useless to the model. In fact, they are essential to the correct learning of the argument structures, since they are the parameters that enable the model to differentiate arguments from adjuncts.

We describe our training data in the next section.

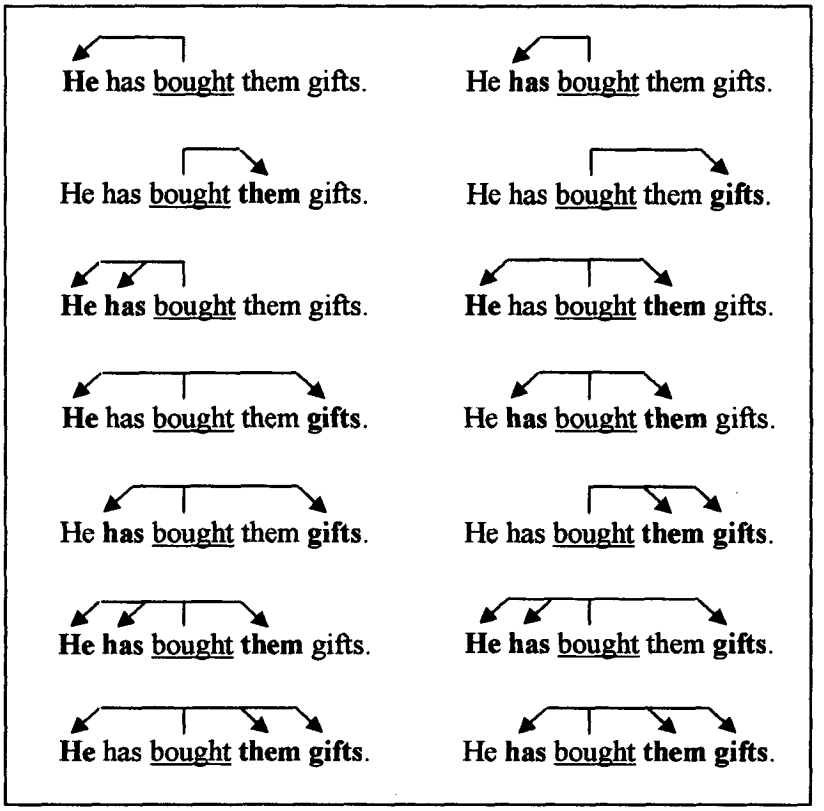


Figure 6. Possible argument structures in the sentence "He has bought them gifts"

5 Data preparation

We have randomly selected 20 verbs – "ask", "begin", "believe", "buy", "cause", "change", "die", "earn", "expect", "find", "give", "help", "kill", "like", "move", "offer", "pay", "raise", "sell", "spend" – for training our model. We extracted from the TREC 2002 corpus all sentences containing occurrences of these verbs. Since our model is too simple to properly cope with long sentences (especially those that contain verb sentential complements), we filtered out the sentences longer than 10 words. We tagged every sentence using the BBN IdentiFinder system (Bikel et al., 1999) and Ratnaparki's (1996) part of speech tagger. The first system is the named entity tagger we use.

The use of a named entity tagger is not necessary for our model to work; however, we thought it would be wise to employ such a tool in order to learn more general argument structures. If entities are not used, the structures we

learn are completely lexicalized; if a named entity tagger is used, we expect to learn both lexicalized and generalized verb argument structures. As expected, named entities overcome some of the data sparseness problems and yield argument structures that are more likely than the fully lexicalized ones. It is worth noting that the most appropriate level of abstraction for arguments (lexemes vs. named entities) is learned automatically by the EM algorithm.

Other arrangements we did to our data include:

- All numbers were replaced by the general entity number.
- Excepting "it", "they" and "them", all personal pronouns were replaced by the entity person; "it", "they" and "them" were considered to be both person and the generic entity thing (that can be anything but person), since they can refer to anything.

Figure 7 shows a sample of our learning data, with entities in bold. It is easy to note that some sentences are completely lexicalized, without entities (e.g., the last sentence), while others have several entities.

about/IN **money**/NN home/NN water/NN heaters/NNS are/VBP bought/VBN each/DT year/NN

person/PRP bought/VBD **thing**/PRP **number**/CD years/NNS ago/RB

organization/NNP bought/VBD **organization**/NN from/IN **organization**/NN last/JJ year/NN

thing/PRP bought/VBD the/DT outstanding/JJ shares/NNS on/IN **date**/NNP

the/DT cafeteria/NN bought/VBD extra/JJ plates/NNS

Figure 7. Data sample

In the first sentence, one can also note an error introduced by the named entity tagger: "8 million" was misclassified as **money** (such errors should be naturally discarded by EM as valid arguments, since they are not frequent in our corpus).

The first two columns of Table 2 show the selected verbs and the number of collected sentences for each of them. On average, we ended up with 2402 sentences per verb and a total of 435.000 words in the collected corpus.

6 Evaluation and analysis

We trained our model on each of the 20 selected verbs. To assess the correctness of the verb argument structures we learned automatically, we carried out two experiments.

For three of the 20 verbs (randomly selected), we kept only the most probable 10 and 20 argument structures learned by our algorithm. We presented these argument structures to three judges and asked them to independently judge their correctness/plausibility. Table 1, which summarizes the results of this experiment, shows that more than 90% of the top-10 induced verb argument structures were judged as correct, as well as approximately 90% of the top-20 structures. The annotation agreement between human judges was also high: the kappa statistic (Carletta, 1996) was 0.77. (A kappa figure between 0.6 and 0.8 indicates high agreement.)

Table 1. Correctness/precision of top-n induced verb argument structures

Verb	**Judge 1**		**Judge 2**		**Judge 3**	
	Top 10	Top 20	Top 10	Top 20	Top 10	Top 20
ask	89	89	89	89	89	95
buy	90	95	100	100	90	89
cause	89	87	89	87	100	94

In order to obtain a more comprehensive understanding of the strengths and weaknesses of our method and its relation to manually produced structures, we also carried out a secondary evaluation, comparing our results to PropBank, one of the large scale projects regarding the semantic specification of verbs. The toughest judge among those who participated in the first experiment (Judge 1) evaluated the correctness of *all* verb argument structures that had a probability higher than a threshold of 10^{-6} and that were supported by at least three sentences in the training corpus. Table 2 summarizes the correctness of all these structures, across all verbs, in terms of both recall and precision. Precision indicates how many of the learned argument structures for a verb were judged plausible by the human, while recall indicates how many of the PropBank possible structures for a verb were learned by our model. During precision calculation, if the judge had any doubt about the correctness of the induced verb argument structure, the judge had access to the supporting sentences that triggered the creation of that structure. In recall calculation, for each possible structure in PropBank, the judge searched for a corresponding learned structure, which could be lexicalized (when there was a match between structures words) or not (when the entities from the learned structure corresponded to the types of the PropBank structures words). Under

these evaluation conditions, our learning algorithm achieved 76% precision and 86% recall. The third column in Table 2 lists the total number of argument structures evaluated for each verb. On average, we learned about 140 structures/verb.

This same evaluation was carried out by the same judge for the top-10 and top-20 argument structures for all verbs. Table 3 shows the average results for each one. As expected, one can note that, as more argument structures we consider, precision decreases and recall increases. Precision decreases because we consider more low probability structures in evaluation; recall increases because we consider more structures, which makes matches with PropBank more probable.

Table 2. Performance of verb argument structure induction algorithm

Verb	Num. of sentences	Num. of argument structures	Precision & recall (%)
ask	3179	212	P=83, R=100
begin	4042	166	P=81, R=50
believe	910	45	P=87, R=100
buy	1106	75	P=79, R=80
cause	957	23	P=74, R=100
change	2648	152	P=88, R=100
die	4154	257	P=72, R=100
earn	952	72	P=83, R=100
expect	6039	422	P=73, R=75
find	4679	210	P=82, R=100
give	3581	249	P=67, R=100
help	1663	53	P=75, R=40
kill	3253	240	P=54, R=100
like	968	74	P=69, R=100
move	2118	162	P=71, R=60
offer	1599	59	P=85, R=67
pay	2076	142	P=76, R=88
raise	1268	79	P=71, R=50
sell	1538	98	P=81, R=100
spend	1322	62	P=61, R=100
Avg.	**2402**	**142**	**P=76, R=86**

Table 3. Performance of the algorithm for top-10 and top-20 verb argument structures

Structures	Precision (%)	Recall (%)
Top-10	93	36
Top-20	89	46
All	76	86

We investigated what led to both recall and precision problems. The recall problem, i.e., the inability of the algorithm to induce certain ProbBank structures, is explained by the following reasons:

- Some of the words found in PropBank structures were not part of our training corpus, and therefore, they were not learned. Our model did predict the corresponding PropBank structure, but with some word variation. For instance, our model learns the structure offer(organization,deal), but not the corresponding Propbank structure, offer(organization,package), because the word "package" is not in our training corpus.
- A few PropBank structures had more than three arguments, because they included adjuncts; our model predicts at most 3 arguments. For instance, PropBank lists the sentence "John killed Mary with a pipe in the conservatory", for which the words "John", "Mary", "pipe" and "conservatory" are arguments, while our model predicts structures with two entities of type person as arguments and a third argument being the instrument or the location of the event, but not both together.

In some cases, we counted these cases as correct when there were learned structures compatible with the PropBank structures, but not in exact matches: in the first case, we permitted some word variation in the learned structures; in the second case, because adjuncts are optional, we ignored the ones predicted by PropBank.

There were two main reasons that explained our precision problems; they pertained to improper handling of adverbs and phrasal verbs. Adverbs should be adjuncts, instead of arguments, and, therefore, should not be included in argument structures. However, in some cases, the adverbs are too frequent and look essential to the sentence meaning, like in "He asked rhetorically" and "He asked incredulously". Corroborating this, PropBank includes adverbs in some argument structures. Phrasal verbs are also a nuisance to our model. For instance, from the sentence "He gave up", the model learns that either "up" is a possible argument for "gave" or that "gave" asks for 1 argument only; both options are inadequate.

For exemplifying the learned structures, Figure 8 shows the top 10 structures learned for the verb "buy". It is worth noting the following:

- the 5th and 6th structures are very similar (in the former, a person buys an organization; in the latter, an organization is bought by a person);
- the 7th structure has a lexicalized item ("house");
- in the 8th structure, there is an error caused by the inclusion of an adverb ("anyway") in the structure (because it co-occurred many times with the verb "buy" in the corpus);
- in the 9th structure, there is an error caused by the phrasal verb "buy down" (like in "dollar bought down the yen") (because the system is not able to identify this as a phrasal verb and ignore it).

For several verbs, our model was able to learn senses and behaviors not listed in PropBank. For instance, for the verb "raise", our model learned structures for the 'growing' sense of the verb (like in "Peter was raised in a big city"), which is not annotated in PropBank. Our model could also learn many possible alternations (not listed in PropBank) for the verb "die", for instance:
(a) In *date*, *person* died.
(b) *Person* died in *date*.
(c) *Person* died in *date* in *location*.
(d) *Person* died in *location* in *date*.

1 buy(organization,organization)	1.20e-01
2 buy(person,number)	8.44e-02
3 buy(person,thing)	7.10e-02
4 buy(organization,thing)	5.63e-02
5 buy(person,organization)	4.28e-02
6 buy(organization,person)	3.51e-02
7 buy(person,house)	1.54e-02
8 buy(person,thing,anyway)	1.54e-02
9 buy(money,money)	1.40e-02
10 buy(organization,organization,date)	8.63e-03

Figure 8. Argument structures for the verb "buy"

7 Discussion

Building on a strong history (Carroll and Charniak, 1992; Pereira and Schabes, 1992; Brill 1993), the unsupervised induction of syntactic structures has recently received increased attention (van Zaanen 2000; Clark, 2001; Klein and Manning, 2004). To our knowledge, our work is the first that tackles the problem of unsupervised learning of semantic representations, i.e., verb argument structures. The experiments reported in the paper make explicit the strengths and weaknesses of our approach. On the positive side, our model

is able to yield high accuracy verb argument structures with no annotation effort, using relatively simple language tools. Our model learns both abstract argument structures, which are grounded in named entity types, and specific structures, which are grounded in the lexicon. Not only does our method find most of the verb argument structures that are already annotated in the PropBank, but it is also able to suggest structures that are not part of this resource.

On the negative side, our model is still too simple. It does not know how to handle phrasal verbs and operates with very simple models of word movement and dependency. Due to its flatness, it cannot work well in conjunction with long sentences. Like FrameNet and VerbNet, our model does not explicitly differentiate between complements and adjuncts.

We believe that using named entity types as abstractions/labels for arguments is both a blessing and a curse. On one hand, one can easily imagine how such structures can be used in a variety of natural language generation applications (summarization and machine translation) to assess whether the generated outputs are consistent with a set of pre-learned structures. If such a system generates text that subsumes an inconsistent structure, that text is probably semantically ill-formed. Having named entities and lexemes as labels for the arguments is adequate in such applications because both named entities and lexemes are accessible to modern summarization and machine translation systems. On the other hand, for applications that require deeper representations of meaning, the level of abstraction specific to the structures we learn may be too lexically grounded and, therefore, inadequate.

Each of the shortcomings discussed above are clear opportunities for future research. In order to enable the development of applications that utilize a resource such as the one described in this paper in NLP applications, the authors are currently working towards automatically inducing the argument structures for the most frequent 2000 English verbs.

Acknowledgments

The authors are grateful to FAPESP (*Fundação de Amparo à Pesquisa do Estado de São Paulo*), CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*), CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*), and Fulbright Comission, which have supported this work. This research was developed at University of Southern California/Information Sciences Institute, under Dr. Daniel Marcu supervision, to whom the authors are also grateful.

References

Baker, C.F.; Fillmore, C.J.; Lowe, J.B. (1998). The Berkeley FrameNet project. In the *Proceedings of COLING/ACL*, pp. 86-90, Montreal.

Bikel, D.M.; Schwartz, R.; Weischedel, R.M. (1999). An Algorithm that Learns What's in a Name. *Machine Learning* (Special Issue on NLP).

Brent, M.R. (1991). Automatic acquisition of subcategorization frames from untagged text. In the *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pp. 209-214, Berkeley, CA.

Brill E. (1993). Automatic grammar induction and parsing free text: A transformation-based approach. In the *Proceedings of ACL*, pp. 259-265.

Briscoe, T. and Carroll, J. (1997). Automatic extraction of subcategorization from corpora. In the *Proceedings of the 5th ANLP Conference*, pp. 356-363, Washington, D.C.

Carletta, J. (1996). Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, Vol. 22, N. 2, pp. 249–254.

Carroll G. and Charniak E. (1992). Two experiments on learning probabilistic dependency grammars from corpora. In the *Working Notes of the Workshop on Statistically Based NLP Techniques*, pp. 1-13.

Clark A. (2001). Unsupervised induction of stochastic context-free grammars using distributional clustering. In the *Proceedings of The Fifth Conference on Natural Language Learning*.

Dempster, A.P.; Laird N.M.; Rubin, D.B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, Ser B, Vol. 39, pp. 1-38.

Fellbaum, C. (1999). *Wordnet: an electronic lexical database*. The MIT Press. Cambridge.

Framis, F.R. (1994). An experiment on learning appropriate selection restrictions from a parsed corpus. In the *Proceedings of the International Conference on Computational Linguistics*, Kyoto, Japan.

Gildea, D. (2002). Probabilistic Models of Verb-Argument Structure. In the *Proceedings of the 17th International Conference on Computational Linguistics*.

Gomez, F. (2004). Building Verb Predicates: A Computational View. In the *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pp. 359-366, Barcelona, Spain.

Green, R.; Dorr, B.J.; Resnik, P. (2004). Inducing Frame Semantic Verb Classes from WordNet and LDOCE. In the *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pp. 375-382, Barcelona, Spain.

Grishman, R. and Sterling, J. (1992). Acquisition of selectional patterns. In the *Proceedings of the International Conference on Computational Linguistics*, pp. 658-664, Nantes, France.

Grishman, R. and Sterling, J. (1994). Generalizing Automatically Generated Selectional Patterns. In the *Proceedings of the 15th International Conference on Computational Linguistics*, Kyoto, Japan.

Kingsbury, P. and Palmer, M. (2002). From Treebank to PropBank. In the *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, Las Palmas.

Kipper, K.; Dang, H.T.; Palmer, M. (2000). Class-based Construction of a Verb Lexicon. In the *Proceedings of AAAI 17th National Conference on Artificial Intelligence*. Austin, Texas.

Klein D. and Manning C. (2004). Corpus-based induction of syntactic structure: models of dependency and constituency. In the *Proceedings of ACL*.

Levin, B. (1993). *Towards a lexical organization of English verbs*. Chicago University Press, Chicago.

Manning, C. (1993). Automatic acquisition of a large subcategorization dictionary from corpora. In the *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pp. 235-242, Columbus, Ohio.

Manning, C.D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

Marcus, M.; Santorini, B.; Marcinkiewicz, M.A. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, Vol. 19, N. 2, pp. 313-330.

Marcus, M. (1994). The Penn Treebank: A revised corpus design for extracting predicate-argument structure. In the *Proceedings of the ARPA Human Language Technology Workshop*, Princeton, NJ.

McCarthy, D. (2000). Using semantic preferences to identify verbal participation in role switching alternations. In the *Proceedings of the 1st NAACL*, pp. 256-263, Seattle, Washington.

Pereira, F. and Schabes, Y. (1992). Inside-outside re-estimation from partially bracketed corpora. In the *Proceedings of ACL*, pp. 128-135.

Ratnaparki, A. (1996). A Maximum Entropy Part-Of-Speech Tagger. In the Proceedings of the Empirical Methods in Natural Language Processing Conference, University of Pennsylvania.

Resnik, P. (1992). Wordnet and distributional analysis: a class-based approach to lexical discovery. In the *Proceedings of AAAI Workshop on Statistical Methods in NLP*.

Rooth, M.; Stefan, R.; Prescher, D.; Carroll, G.; Beil, F. (1999). Inducing a semantically annotated lexicon via EM-based clustering. In the *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 104-111, College Park, Maryland.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, Vol. 27, N. 3, pp. 379-423.

van Zaanen M. (2000). ABL: Alignment-based learning. In the *Proceedings of COLING*, pp. 961-967.