Lexical Attraction Models of Language

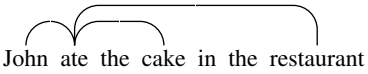
Deniz Yuret

Koç University, Istanbul, Turkey, dyuret@ku.edu.tr

Abstract. This paper presents on-going work on the lexical attraction models of language, in which the only explicitly represented linguistic knowledge is the likelihood of pairwise relations between words. This is in contrast with models that represent linguistic knowledge in terms of a lexicon, which assigns categories to each word, and a grammar, which expresses possible combinations in terms of these categories. The word-based nature and the simplicity of lexical attraction models make them good candidates for experiments in language learning. I introduce a parser and an on-line unsupervised learning algorithm based on expectation maximization.

1 Introduction

Every sentence in natural language has an underlying structure that consists of the relations between its words:



This simple representation indicates the related words with links connecting them, i.e. *John* is the subject and *cake* is the object of the verb *ate*, *restaurant* is where the eating took place. If the sentence were *"John ate the cake in the box."*, the word *box* would be linked to *cake* because presumably John was not in the box while eating. We effortlessly make such inferences because we know cakes usually come in boxes and people usually eat in restaurants.

Word associations such as *eat-restaurant* and *cake-box* are not only useful for resolving syntactic ambiguity, but they may play an important role in early child language acquisition. A common assumption made by language acquisition theories is that children understand word meanings before syntax acquisition begins. Word meanings coupled with general world knowledge is usually sufficient to identify word pairs in the sentence that are related. When a pre-syntax child hears *"John ate the cake"*, he can conclude that *John* is the eater and *cake* is the thing that is eaten regardless of the positions of the words. This conclusion in turn fuels the acquisition of syntactic patterns employed by his particular language to express functions such as subject and object.

A computer can acquire such word associations from a large corpus. It can then use this information to infer relations between words in a sentence. The nature and accuracy of these relations are of interest because they represent an important portion of the information available to the pre-syntax child.

Lexical attraction can be defined as the likelihood of two words being syntactically related in a sentence. When deciding whether two words are related in a sentence we use two types of information. First, there are grammatical constraints, e.g. each determiner must modify a noun or transitive verbs must take objects. Second, there are selectional restrictions, e.g. given the verb *eat*, *cake* would be a more likely object than *book*. Grammatical constraints by themselves are typically not restrictive enough to uniquely identify the correct relations. Lexical attraction models quantify selectional restrictions and can be used to demonstrate how much of the sentence we can understand without the help of syntax.

Linguistic models based on pairwise relations between words are called dependency models, and have a long history[1]. Sleator et. al.[2] developed one of the first large scale implementations of dependency grammar for English. Supervised learning of probabilistic dependency models was introduced in [3, 4]. Unsupervised learning for dependency models was introduced in [5] and later developed by [6]. More recently, unsupervised learning using both dependency and constituency was explored in [7].

Section 2 gives some basic results on dependency structures. Section 3 formalizes the lexical attraction model in the language of information theory. Section 4 describes the parsing algorithm. Section 5 presents the unsupervised learning algorithm and experiments.

2 Dependency Structures

The linguistic formalism that takes syntactic relations between words as basic primitives is known as the dependency formalism. Mel'čuk discusses important properties of syntactic relations in his book on dependency syntax [1]. Sleator and Temperley have a large scale implementation of English syntax based on a similar formalism they call link grammars [2]. This section presents basic properties of linguistic dependency structures.

In this work we will assume that the dependency structure is acyclic. It is generally the case that the syntactic relations in a sentence form an acyclic graph, i.e. a tree. Linguistically, each word in a sentence has a unique head, except for the root word, which governs the whole sentence[1].

Most sentences in natural languages also have the property that syntactic relation links drawn over words do not cross. This property is called *planarity* [13], *projectivity* [1], or *adjacency* [14] by various researchers. The examples in Figure 1 illustrate the planarity of English. In the first sentence, it is easily seen that the woman was in the red dress and the meeting was in the afternoon. However, in the second sentence, the same interpretation is not possible. In fact, it seems more plausible for John to be in the red dress.

Gaifman gave the first formal analysis of dependency structures that satisfy the planarity condition[15]. His paper gives a natural correspondence between

[1] See [1, p. 25] for a discussion.

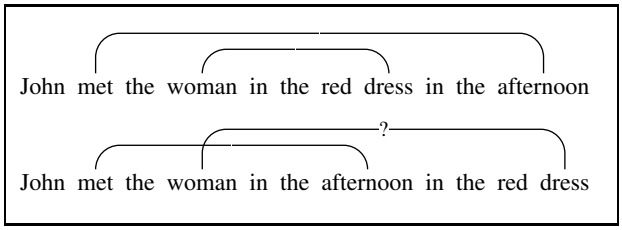


Fig. 1. Example illustrating planarity

dependency systems and phrase-structure systems and shows that the dependency model characterized by planarity is context-free. Sleator and Temperley show that their planar model is also context-free even though it allows cycles [2].

The number of possible dependency structures for an n word sentence is given by $f(n) = C(3n-1, n-1)/(2n-1)$, where C indicates the binomial coefficient. The first few values of $f(n)$ are: 1, 1, 3, 12, 55, 273, 1428. Figure 2 shows the possible planar dependency structures with up to four words.

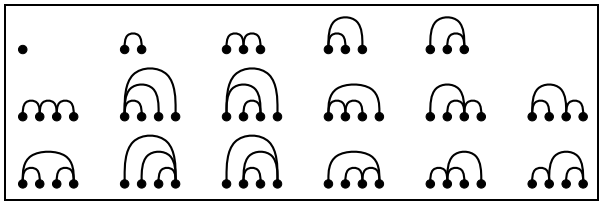


Fig. 2. Possible planar dependency structures with up to four words.

3 The Lexical Attraction Model

Lexical attraction is best described within the framework of information theory. Shannon defines the entropy of a discrete random variable as $H = -\sum p_i \log p_i$ where i ranges over the possible values of the random variable and p_i is the probability of value i [8, 9]. Consider a sequence of tokens drawn independently from a discrete distribution. In order to construct the shortest description of this sequence, each token i must be encoded using $-\log_2 p_i$ bits on average. $-\log_2 p_i$ can be defined as the information content of token i. Entropy can then be interpreted as the average information per token. Following is an English sentence with the information content of each word given below, assuming words are independently selected. The word probabilities were estimated using a large

corpus of news material. Note that the information content is lower for the more frequently occurring words.

The	IRA	is	fighting	British	rule	in	Northern	Ireland
4.20	15.85	7.33	13.27	12.38	13.20	5.80	12.60	14.65

3.1 Mutual information

Language models achieve lower entropy by taking into account the relations between the words in the sentence. Consider the phrase *Northern Ireland*. Even though the independent probability of *Northern* is $2^{-12.6}$, it is seen before *Ireland* 36% of the time. Another way of saying this is that although *Northern* carries 12.6 bits of information by itself, it adds only $\log_2(0.36) = 1.48$ bits of new information to *Ireland*.

With this dependency, *Northern* and *Ireland* can be encoded using $1.48 + 14.65 = 16.13$ bits instead of $12.60 + 14.65 = 27.25$ bits. The 11.12 bit gain from the correlation of these two words is called mutual information. We measure lexical attraction with mutual information. The basic assumption of this work is that words with high lexical attraction are likely to be syntactically related.

The following diagram gives the information content of the words in our example according to a bigram model. The information content of each word is computed based on its conditional probability given the previous word. As a result, the encoding of the sentence is reduced from 99.28 bits to 62.34 bits.

The	IRA	is	fighting	British	rule	in	Northern	Ireland
4.20	12.90	3.73	10.54	8.66	5.96	3.57	9.25	3.53

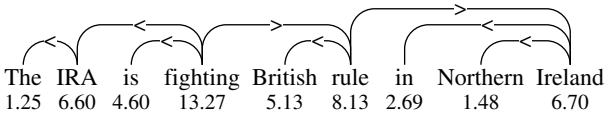
3.2 Linguistic context

Using the previous word as context is against our linguistic intuition. In a sentence like *"The man with the dog spoke"*, the selection of *spoke* is determined by *man* and is independent of the previous word *dog*. It follows that the context of a word would be better determined by its linguistic relations rather than according to a fixed pattern.

The assumption in lexical attraction models is that each word depends on one other word in the sentence, but not necessarily an adjacent word as in n-gram models. Lexical attraction models make it possible to define the context of the word in terms of its syntactic relations.

Words in direct syntactic relation have strong dependencies. Chomsky defines such dependencies as *selectional relations* [11]. Subject and verb, for example, have a selectional relation, and so do verb and object. Subject and object, on the other hand, are assumed to be chosen independently of one another. It should be noted that this independence is only an approximation. The sentences *"The doctor examined the patient"* and *"The lawyer examined the witness"* show that the subject can have a strong influence on the choice of the object.

The following diagram gives the information content of the words in the example sentence based on direct syntactic relations:



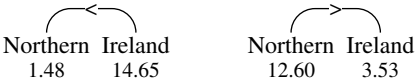
The arrows represent the head-modifier relations between words. The information content of each word is computed based on its conditional probability given its head. I marked the verb as governing the auxiliary and the noun governing the preposition which may look controversial to linguists. From an information theory perspective, the mutual information between content words is higher than that of function words. Therefore the model does not favor function word heads.

The probabilities were estimated by counting the occurrences of each pair in the same relative position. The linguistic dependencies reduce the encoding of the words in this sentence to 49.85 bits compared to the 62.34 bits of the bigram model. However, note that this number excludes the encoding of the dependency structure.

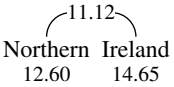
3.3 Symmetry of lexical attraction

Lexical attraction between two words is symmetric. The mutual information is the same no matter which direction the dependency goes. This directly follows from Bayes' rule. What is less obvious is that the choice of the head word and the corresponding dependency directions it imposes do not effect the joint probability of the sentence. The joint probability is determined only by the choice of the pairs of words to be linked.

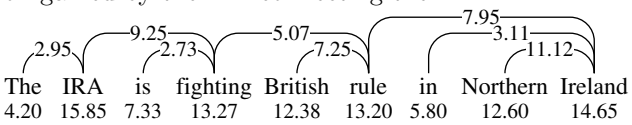
Consider the *Northern Ireland* example:



In the first case, I used the conditional probability of *Northern* given that the next word is *Ireland*. In the second case, I used the conditional probability of *Ireland* given that the previous word is *Northern*. In both cases the encoding of the two words is 16.13 bits, which is in fact $-\log_2 p$ of the joint probability of *Northern Ireland*. Thus a more natural representation would be the following, where the link has no direction and its label shows the number of bits gained, i.e. mutual information:



I generalize this result below and use the same representation for the whole sentence. The lexical attraction for a pair of words is defined as the mutual information gained by the link connecting them.



4 The Parsing Algorithm

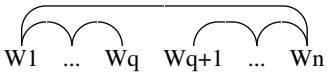
In this section we will describe an algorithm that finds the dependency structure with the highest probability for a given sentence and lexical attraction model. The straightforward way to do this would involve the comparison of exponentially many structures. A more efficient procedure exists and can perform the calculation in $O(n^3)$ steps for an n word sentence.

The leftmost word w_1, in the best possible structure, may be linked to several other words in the sentence, but there must be one that is furthest to the right. Assume that word is w_r.

Consider the case when $r \neq n$. Then the words between w_1 and w_r cannot link to the words after w_r, because that would break the no-link-crossing principle. w_1 also cannot link to a word after w_r because w_r was presumed to be the final word linked to w_1.

We have divided the sentence into two parts that overlap by one word, w_r. If we multiply the lexical attraction products for the two parts, we get the product for the whole sentence. Each part has fewer words than the whole sentence, so the recursion is guaranteed to stop. So, at this point, it would be a matter of trying all r, and picking the best, except there is the possibility that w_1 does link to w_n.

Consider the case when w_1 is linked to w_n. There must be some words between w_1 and w_n that ultimately link to w_1 and others that ultimately link to w_n. The two groups must be disjoint, otherwise the dependency structure would be cyclic. Suppose w_q is the rightmost word that links ultimately to w_1.



We have divided the sentence into two parts $w_1 \ldots w_q$ and $w_{q+1} \ldots w_n$, with a single link joining the two parts. If we multiply the lexical attraction products for the two parts, and the extra factor for the $w_1 w_n$ link, we get the product for the whole sentence.

Let us call the dependency structures where w_1 is linked to w_n *covered* and the others *uncovered*. We can find the best dependency structure by checking the covered and the uncovered structures and picking the best one.

The computation can be performed bottom up by starting with short segments of the sentence and inductively computing the best structures for longer segments. The base of the induction will be consecutive word pairs, for which linking the two words is the only possible structure. We will use the following notation.

D_{ij} = the best dependency structure for the segment $w_i \ldots w_j$.
D^c_{ij} = the best covered structure for $w_i \ldots w_j$.

D^u_{ij} = the best uncovered structure for $w_i \ldots w_j$.
$\rho(D)$ = the total lexical attraction for D.
$D_1 \cup D_2$ = union of the links in two dependency structures.
$\langle i, j \rangle$ = a link between w_i and w_j.

D_{ij} can be inductively computed as follows.

1. $r = \arg\max_k \rho(D^c_{ik}) + \rho(D_{kj})$
2. $D^u_{ij} = D^c_{ir} \cup D_{rj}$
3. $q = \arg\max_k \rho(D_{ik}) + \rho(D_{k+1j})$
4. $D^c_{ij} = D_{iq} \cup D_{q+1j} \cup \{\langle i, j \rangle\}$
5. $D_{ij} = \arg\max_D \rho(D^u_{ij}), \rho(D^c_{ij})$

The first two steps compute the best uncovered structure. Note that the first part D^c_{ir} is covered because r is linked to i. The next two steps compute the best covered structure. Note that the lexical attraction factor for the $w_i w_j$ link is not needed in step 3 because it is fixed for all partitions. The final step chooses between the two best structures.

In order to compute D_{1n}, these steps will have to be performed for each i, j pair where $1 \leq i < j \leq n$. Steps 1 and 3 require $O(n)$ steps to find the best partition. Therefore the algorithm requires $O(n^3)$ steps to find D_{1n}, the best structure for the sentence.

5 Learning Experiments

This section presents an on-line unsupervised learning algorithm and the results of training lexical attraction models. The algorithm interdigitates learning and processing for every input sentence in the following manner:

- Find the most likely structure for an input sentence given the current state of the model.
- Update the model assuming the structure found is the correct one.

In the following experiments, test results were obtained using one million words of annotated Wall Street Journal text from the Penn Treebank corpus [19]. The phrase structure annotations were converted into dependency structure links and the results give the percentage of links guessed correctly in the experiment. For unsupervised training, out of sample plain WSJ text was used. Experiments 1–3 were run for benchmarking purposes.

Experiment 1 In this experiment, the sentences were parsed using a random model. The model used a random number generator to generate the lexical attraction values. The resulting accuracy was 25.2%.

Experiment 2 The annotated test data was used for supervised training. The resulting model was tested on the same data to get an upper bound on the performance of a lexical attraction model which resulted in 77.4% accuracy.

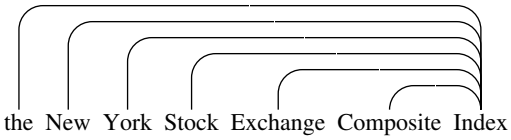
Experiment 3 One thousand sentences were held out from the test data and the remaining was used for supervised training. Testing on the held out data led to 53.4% accuracy.

Experiment 4 In this experiment, I started with an empty model, giving 0 frequency for any word pair. I used the same 20 million words of training data as the previous experiment. The parser was run on every sentence using the existing model. The model was updated after every sentence by incrementing the frequencies of the linked pairs. The resulting accuracy was 35.5%.

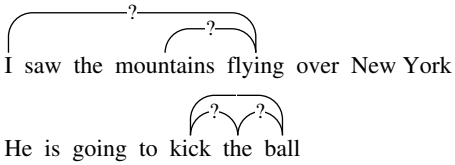
Experiment 5 Starting with an empty model and using the same 20 million words of training data, the parser was run on every sentence using the existing model as in the previous experiment. However, the model was updated after every sentence by incrementing the frequencies of all first and second degree neighbors. The resulting accuracy was 40.4%. The accuracy for content word links was 51.9%.

The difference between experiments 4 and 5 is significant. When the model is empty, giving 0 frequency to each pair, the parser ends up linking adjacent words. In experiment 4, positive mutual information between some linked pairs is discovered and the parser starts linking them even when they are not adjacent. However, related words that are never seen adjacent cannot be discovered. These include objects of verbs separated by determiners, as in "**kick** the **ball**", or prepositional phrases and adverbials modifying a verb that need to follow the object, such as "**kick** the ball **today**". Incrementing the second degree neighbors in experiment 5 solves this problem and lets the program discover all related words eventually.

The use of Penn Treebank data for evaluation of unsupervised language acquisition is debatable. The conversion from the phrase structure representation to dependency structure representation is bound to have errors. These errors are most pronounced in noun phrases, where the Penn Treebank gives no internal structure. The heuristics employed lead to the following linkage which results in an accuracy result of 50% for the correct linkage discovered by the program:



Another problem with the evaluation is the equal treatment of content and function words. For extraction of meaning, the mistakes in content-word links are significantly more important than the mistakes in function-word links. The following two sentences illustrate the difference. In the first sentence, a mistake would result in choosing the wrong subject for flying. In the second sentence, once the program has detected the relation between kick and ball, which way the word *the* links is less important.



The program used in experiment 5 discovers 45.6% of the content-word links correctly. With 180 million words of training, this accuracy goes up to 51.9%.

6 Contributions

I presented some basic results for lexical attraction models, in which the only explicitly represented linguistic knowledge is the likelihood of pairwise relations between words. I showed that these models can be used for unsupervised language learning. Lexical attraction can perform well in situations where grammatical constraints are not sufficiently restrictive, such as the analysis of complex noun phrases, prepositional phrase attachment, and general syntactic ambiguity. Figure 3 shows example sentences taken from various stages of unsupervised learning with ten million words of news material.

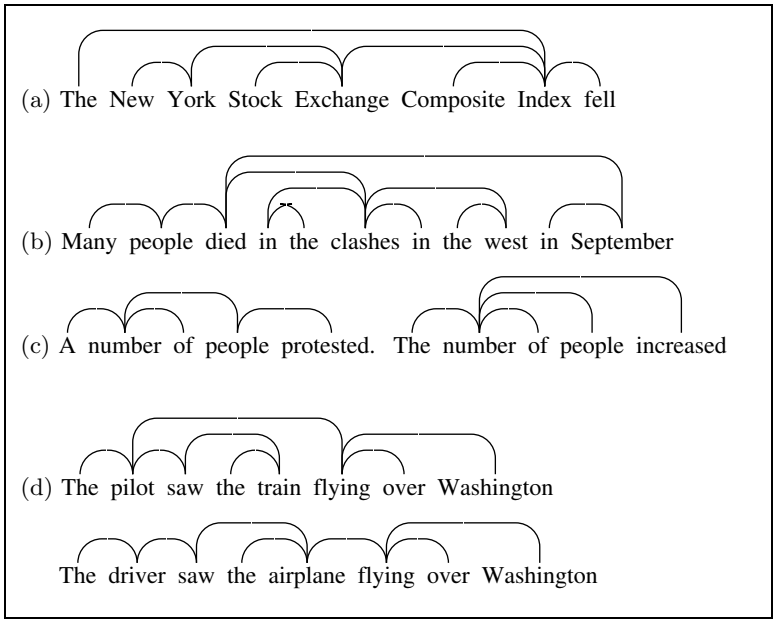


Fig. 3. Example sentences output by the model that are particularly challenging for syntactic models: (a) the internal structure of a complex noun phrase, (b) prepositional phrase attachment, (c) and (d) two examples of syntactic ambiguity.

References

1. Mel'čuk, I.A.: Dependency syntax: theory and practice. SUNY (1988) inci.
2. Sleator, D., Temperley, D.: Parsing english with a link grammar. Technical Report CMU-CS-91-196, CMU (1991)
3. Lafferty, J., Sleator, D., Temperley, D.: Grammatical trigrams: a probabilistic model of link grammar. In: AAAI Fall Symposium on Probabilistic Approaches to Natural Language Processing. (1992)
4. Carroll, G., Charniak, E.: Learning probabilistic dependency grammars from labeled text. In: Probabilistic Approaches to Natural Language, Papers from 1992 AAAI Fall Symposium. (1992) 25–31
5. Yuret, D.: Discovery of linguistic relations using lexical attraction. PhD thesis, MIT (1998)
6. Paskin, M.A.: Grammatical bigrams. In Dietterich, T., Becker, S., Gharahmani, Z., eds.: Advances in Neural Information Processing Systems 14 (NIPS-01), Cambridge, MA, MIT Press (2001)
7. Klein, D., Manning, C.: Corpus-based induction of syntactic structure: Models of dependency and constituency. In: Proceedings of the 42nd Annual Meeting of the ACL. (2004)
8. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal **27** (1948)
9. Cover, T.M., Thomas, J.A.: Elements of information theory. John Wiley and Sons, Inc. (1991)
10. Beeferman, D., Berger, A., Lafferty, J.: Text segmentation using exponential models. In: EMNLP-2. (1997)
11. Chomsky, N.: Aspects of the theory of syntax. MIT Press (1965)
12. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann (1988)
13. Sleator, D., Temperley, D.: Parsing english with a link grammar. In: Third international workshop on parsing technologies. (1993)
14. Hudson, R.A.: English word grammar. Blackwell (1990)
15. Gaifman, H.: Dependency systems and phrase-structure systems. Information and Control **8** (1965) 304–337
16. Harary, F.: Graph theory. Addison-Wesley (1969)
17. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete mathematics. 2 edn. Addison-Wesley (1994)
18. Eisner, J.M.: Three new probabilistic models for dependency parsing. In: COLING-96. (1996)
19. Marcus, M.P., et al.: The penn trebank: Annotating predicate argument structure. In: ARPA Human Language Technology Workshop. (1994)
20. Chelba, C., Jelinek, F.: Exploiting syntactic structure for language modeling. In: ACL 98. (1998)