Prof. Andrew Y. Ng
CS 229: Machine Learning
Autumn 2008-2009

Advisor: Prof. Dan Jurafsky, Stanford University;
Supervisor: Dr. Hiyan Alshawi, Google Inc.

Valentin I. Spitkovsky
Final Project
2008-12-12

Unsupervised Learning of Hierarchical Dependency Parsing

1 Very Brief Introduction, Background and Motivation

In a sentence, a dependency is a link from one word, called a dependent or *attachment*, to another, the *head*. If we imagine that the global head word itself connects to some root note, then a sentence of length n corresponds to a parse tree with exactly n links. Constituency parses, such as those in the *Wall Street Journal* section of the Penn Tree-Bank, can be converted to dependency parses using special "head-percolation" rules. Resulting references are then used for training and/or testing hierarchical dependency parsers; the standard quality metric is the fraction of correctly identified links. The problem of training such parsers — and especially its unsupervised variant (thanks to an abundance of texts from specialized genres and in exotic languages and due to the poverty of manually produced references) — is still a very active area of research. Potential beneficiary applications include machine translation, information retrieval, web search and query refinement, to name a few.

Last quarter, in Prof. Manning's CS224N: Natural Language Processing, we chose to implement Klein & Manning's *Dependency Model with Valence* for our final project. A generative model, the DMV lends itself to unsupervised learning, by means of Expectation Maximization via Inside-Outside re-estimation. It actually took us the entire summer to debug our derivations and code, but in the end our implementation actually out-performed the advertised claims of the DMV, which is itself considered state of the art. In this project, we proceeded to tease apart the inner workings of the DMV, in an attempt to improve and/or simplify wherever possible, making use of the various Machine Learning insights gleaned over the term.

2 Just a Few More Necessary Preliminaries

The DMV: To have any hope of success, instead of dealing with actual words $\{w\}$, the model operates over part-of-speech tags as lexical word classes $\{c_w\}$, of which there are only a few dozen. The generative story for a sub-tree rooted at some head c_h rests on three independence assumptions: 1) c_h first decides whether to attach all of its left or all of its right children first, using the probability $\mathbb{P}_{\texttt{ORDER}}(c_h)$; 2) it then generates children in a particular direction $dir \in \{\texttt{L}, \texttt{R}\}$ until it stops, at each point with probability $\mathbb{P}_{\texttt{STOP}}(c_h, dir, adj)$, where $adj \in \{\texttt{T}, \texttt{F}\}$ (true iff this is the first or *adjacent* child); and 3) each child slot is filled in with an attachment of class c_a, according to $\mathbb{P}_{\texttt{ATTACH}}(c_h, dir, c_a)$. This disallows crossing dependencies (non-projective parse trees) and corresponds to a context-free[1] grammar, in which all sentences are imagined to end with the special token $\diamondsuit$, whose class serves as the global head, attaching exactly one child on its left (and none on its right). All probabilities are estimated without smoothing.

Initialization: As is often the case, this learning algorithm is very sensitive to initialization of parameters. Both $\mathbb{P}_{\texttt{ORDER}}$ and $\mathbb{P}_{\texttt{STOP}}$ are set uniformly to one half; for $\mathbb{P}_{\texttt{ATTACH}}$, however, the model is given a slight nudge in a linguistically sensible direction. Within each sentence, the likelihood of attachment between two words is modeled as inversely proportional to their distance; the resulting empirical distributions are then averaged over all head words from all sentences and used to seed $\mathbb{P}_{\texttt{ATTACH}}$.

Termination: We had to concoct our own terminating heuristic for EM, so we chose to be conservative, bailing out once either of the following conditions become true: 1) the difference between current and best or last average per-link cross-entropy drops below 2^{-30}; 2) number of iterations reaches 1,000.

Data Set: The DMV is trained/tested on WSJ10, a modified subset of the PTB: annotated parse trees are stripped of all empty elements and punctuation (tagged as `,`, `.`, `:`, `-LRB-` and `-RRB-`), as well as items which aren't pronounced[2] (or at least not pronounced where they appear, tagged `#` and `$`), then thrown out if they still contain more than ten terminals. This leaves 7,422 of the original 49,208 sentences.

Metrics: Requiring the parser to produce a single "best" parse for a given sentence, its *directed score* is then the simple fraction of correct links; to facilitate comparison with earlier work, as well as to partially obscure the effects of alternate

[1] This isn't quite a probabilistic context-free grammar in the strict sense: although the grammar aspect is itself context-free, the rewrite probabilities are not, since some are conditioned on adjacencies. And while the original authors may have needed this elaborate (P)CFG, our purposes are more modest. In particular, all of these independence assumptions imply that left children are entirely independent of the right. Thus, we drop $\mathbb{P}_{\texttt{ORDER}}$ altogether, making that choice deterministic (this reduces the state of the estimation problem and eliminates some noise and rounding pollution). Furthermore, we simplify the calculation of our inside probabilities, e.g. that the word-class in position p derives the sequence of word-classes in positions $x, x+1, x+2, \ldots, y-1$, $0 \leq x \leq p < y$, to be the product of probabilities associated with $x, x+1, x+2, \ldots, p, p+1$ and $p, p+1, p+2, \ldots, y-1$, which peals away one layer of a very nested loop, thus speeding up our implementation, while also helping guard against excessive noise and rounding issues.

[2] The original authors argue that unsupervised learning could shed light on language acquisition in children, and so include only what's audible.

analyses (e.g. systematic choice between a modal and a main verb for the head of a sentence), the original authors also make use of the more flattering *undirected score*, which ignores the directionality of each parent-child relation.

Scores: The *Adjacent Word Heuristic*, which simply links each word with its immediate neighbor (either all going left or all going right) yet shatters all previous unsupervised work, achieves a directed score of 33.6% on this data set; the DMV further improves on its authors' own baseline, scoring 43.2% (63.7%) directed and undirected.

3 Alternative Metrics

One problem with best-parse scoring is that it does not lend itself to exact duplication by an independent implementation, because of the many opportunities for idiosyncratic tie-breaking in standard chart-based algorithms. It can also over- or under-state a model's performance, depending on how an implementation's arbitrary but systematic preference among ties fits with a particular language (e.g. one from a left- or right-branching family) being learned. Rather than simply hard-counting each link as either right or wrong, we estimate its probability softly, as the mass of all parse trees in which it is present, according to the model.

Unlike *best*-scoring, *average*-scoring isn't used in the literature — most likely because it is harder to derive and more difficult to compute; however, given the Inside-Outside charts, it is actually marginally cheap and has the added benefit of being entirely deterministic (modulo floating-point rounding errors) and hence reproducible. Furthermore, it can provide extra insights into certain "undifferentiated" models, e.g. those initialized uniformly, in which the number of ties can be exponential.

We append these average scores to the ordinary best scores in square brackets, when available. In addition, we compute the average per-link cross-entropy (in bits) and include this number with each set of scores as well (in the average case, instead of using the probability of the best parse, we sum the probability of all parses to arrive at the probability of the sentence; these are naturally higher, leading to lower perplexities). In this format, our own optimized (but otherwise vanilla) implementation of the DMV framework achieves (on the standard data set) the following base-lines:

D	*U*	*x-H*	*D*	*U*	*x-H*	*Model Parameters*
19.367%	(39.464%)	5.5363	[18.980%	(37.022%)	4.3304]	Undifferentiated
35.194%	(53.434%)	4.0420	[24.593%	(43.895%)	3.3409]	Ad-Hoc Harmonic
48.176%	(64.556%)	2.5170	[47.775%	(64.165%)	2.4824]	Unsupervised
81.647%	(83.888%)	2.7989	[76.294%	(79.340%)	2.6484]	Supervised
Best			*Average*			

These are, respectively, the undifferentiated (all uniform) model; the ad-hoc harmonic (used to initialize unsupervised learning); the result of unsupervised learning; and a supervised (oracle) model, which simply reads off all probabilities from the reference parse trees. What's even more surprising than the fact that our best directed score of 48.2% is significantly higher than the reported 43.2% is that the supervised model's perplexity is worse: task performance does not vary linearly with the model's goodness of fit!

4 Ablative Analysis

Other red flags raised by our baselines include the facts that 1) even the system with perfect knowledge is quite far from 100%; 2) undirected scores for initialization and its unsupervised result are quite close, at least on the best parse metric. These inspired us to proceed with further supervised dissection of the DMV, to see how well various partial oracles perform, without unsupervised training:

81.647%	(83.888%)	2.7989	[76.294%	(79.340%)	2.6484]	Perfect Knowledge
70.439%	(73.182%)	3.6977	[61.550%	(66.965%)	3.3967]	All Attached Knowledge
59.516%	(63.974%)	4.0575	[52.296%	(58.504%)	3.6534]	All Left Knowledge
57.447%	(63.851%)	4.1362	[48.451%	(56.662%)	3.7575]	All Right Knowledge
56.821%	(62.594%)	4.5412	[49.645%	(56.386%)	4.1163]	Adjacent Stop Knowledge
54.680%	(60.655%)	4.4003	[50.446%	(57.916%)	3.9961]	All Stop Knowledge
14.552%	(40.049%)	5.0522	[21.429%	(40.157%)	4.1530]	non-Adjacent Stop Knowledge
35.194%	(53.434%)	4.0420	[24.593%	(43.895%)	3.3409]	Ad-Hoc Harmonic
19.367%	(39.464%)	5.5363	[18.980%	(37.022%)	4.3304]	Zero Knowledge

We continue to sort our results by the best directed score, and already some funny business begins to emerge: knowing the full stopping probabilities appears worse than knowing just the adjacent ones, while knowing just the non-adjacent stopping probabilities appears worse than knowing nothing at all (however, average scores better reflect reality). Still, it appears that the model would perform better, as far as parsing is concerned, if the non-adjacent stopping probabilities were eliminated and replaced with the constant one half (or something even more appropriate). Also worth noting is that this last one is the only partial oracle that's worse than the full unsupervised model — there is much room for improvement! Lastly, for decently performing partial oracles, directed and undirected scores seem to be quite close, which doesn't appear to be the case for badly trained models.

5 Robustness and Scaling Exploration

To get a better feel for the DMV's properties, we trained it (in the standard, unsupervised fashion) on WSJ5 and WSJ15 as well. Keep in mind that these are rather weird, non-comparable juxtapositions, since adding long sentences to an existing data set significantly complicates the learning problem (the possible number of parses is exponential in the length of a sentence), but also offers a non-trivial amount of extra training material (each new sentence provides more data than ones before):

54.457%	(70.515%)	2.4957	[54.264%	(70.248%)	2.4785]	WSJ5	[2,008 sentences]
48.176%	(64.556%)	2.5170	[47.775%	(64.165%)	2.4824]	WSJ10	[7,422 sentences]
49.820%	(64.758%)	2.5114	[48.669%	(63.411%)	2.4398]	WSJ15	[15,922 sentences]

As part of our earlier debugging efforts, we also implemented a Viterbi approximation: instead of going through all the error-prone and computationally expensive trouble of Inside-Outside re-estimation, it learns from the single "best" parse of each sentence, given the current model.

48.889%	(66.395%)	2.6868	[49.244%	(66.469%)	2.6516]	WSJ5	[Viterbi]
53.839%	(65.492%)	2.6880	[53.391%	(64.843%)	2.6346]	WSJ10	"
46.697%	(58.684%)	2.6367	[45.782%	(57.824%)	2.5667]	WSJ15	"

Our other experiments suggest that the DMV's quality degrades gracefully as sentences grow past length 15; Viterbi approximation continues to beat the exact solution on directed scores for longer sentences, though the gap trends downwards.

Meant as a quick sanity-checking hack, Viterbi approximation offers a simpler, faster approach, which frequently beats its slower, more complicated state of the art cousin. Given the alarm bells from before, we strongly suspect that the DMV is not the most appropriate model of natural language: it would appear that Viterbi rushes in the direction that's approximately right, while full-fledged EM/IO drills down towards a solution that's exactly wrong... Consider what happens when we initialize both algorithms with the supervised model's "perfect" oracle, back on WSJ10:

81.647%	(83.888%)	2.7989	[76.294%	(79.340%)	2.6484]	Perfect Knowledge	[Supervised]
79.844%	(82.399%)	2.6808	[77.354%	(79.971%)	2.6061]	Perfectly Seeded	[Viterbi]
69.718%	(74.753%)	2.5737	[69.390%	(74.348%)	2.5233]	Perfectly Seeded	[EM/IO]

Viterbi manages to hang on to the good solution, while EM/IO does not. However, we don't think that this is a bug in our implementation of EM/IO, since the more complex algorithm does manage to arrive at lower perplexity values than its counterparts... Nevertheless, since EM/IO is numerically more intensive, we suspected that perhaps rounding and numerical precision issues were muddying things. We managed to eliminate this possibility by testing both approaches nine ways: 1) using three different level of floating point precision in our numeric representation: {`long double`, `double`, `float`}; and 2) using three different abstract implementations of arithmetic over probabilities: regular p-space [`p`], $\log(p)$-space [`log`] and $\log(p)$-space with "sloppy math"[3] for addition / subtraction [`slop`]. Starting each of the nine combinations with each of the four set-ups, $\{\text{Viterbi}, \text{EM/IO}\} \times \{\text{Ad-Hoc Harmonic}, \text{Perfectly Seeded}\}$, we arrived at amazinlgy similar numbers, though the number of iterations till convergence varied significantly between certain extreme arrangements.

6 Some Quick Improvements

We already pointed out that the handling of non-adjacent stopping probabilities could use some help. As is, the model is quite elegant and recursive, lending itself to Inside-Outside re-estimation. But this is also its down-fall: natural language is not recursive — the probability of attaching a 15th child is not the same as for the 2nd... Modeling the number of children as a Geometric distribution leaves way too much probability mass for the very long syntactic structures. But given that the Viterbi approximation is quite good anyway, we take another step away from context-freedom by conditioning on the number of children (which adds an extra layer to the nested loop of Viterbi's inside chart generation).

One elegant approach is to use a probability distribution that still supports arbitrarily long outcomes, even when what's observed is bounded, as did the Geometric, whose $\mu = p^{-1}$. We managed to find a suitable distribution, called the Log-Series Distribution, whose existence follows from the fact that the Taylor expansion of the natural logarithm around $1-x$, $x \in (0,1)$, is $\sum_{i=1}^{\infty} \frac{x^i}{i}$; the probability of landing on any partcular positive integer is $\frac{p^i}{i}$ and $\mu = \frac{-1}{\ln(1-p)} \cdot \frac{p}{1-p}$. We call this the lDMV. Another approach is to memorize non-adjacent probabilities, conditioned on the number of children already attached, as is done for the adjacent probability, which makes it impossible to have more children (on either side) than one minus the length of the longest sentence. We call this DMVc. Still on WSJ10, though now lacking average scores,

48.176%	(64.556%)	2.5170	[47.775%	(64.165%)	2.4824]	[EM/IO]
53.839%	(65.492%)	2.6880	[53.391%	(64.843%)	2.6346]	[Viterbi]
52.752%	(64.802%)	2.6637				[lDMV]
54.040%	(65.461%)	2.6923				[DMVc]

[3] For instance, instead of exponentiating back to p-space, addition is performed using the fact that

$$\log(x+y) = \max(\log x, \log y) + \log\left(1 + e^{\min(\log x, \log y) - \max(\log x, \log y)}\right).$$

DMVc parses better, while the lDMV has lower perplexity than their predecessors. The gains may not appear very significant at first glance, but these models really shine when a poor set of parameters requires rescuing. Consider what happens when the DMV is trained using undifferentiated seeds (rather than with Ad-Hoc Harmonic), also on WSJ10:

22.931%	(52.735%)	2.6246	[22.420%	(52.174%)	2.5636]	[EM/IO]
17.118%	(48.777%)	2.7371	[17.456%	(48.750%)	2.6664]	[Viterbi]
24.868%	(54.526%)	2.6572				[lDMV]
30.623%	(54.877%)	2.8006				[DMVc]

Here is how they fare on WSJ15, this time seeded with Ad-Hoc Harmonic:

49.820%	(64.758%)	2.5114	[48.669%	(63.411%)	2.4398]	[EM/IO]
46.697%	(58.684%)	2.6367	[45.782%	(57.824%)	2.5667]	[Viterbi]
47.592%	(59.356%)	2.6306				[lDMV]
47.112%	(58.974%)	2.6331				[DMVc]

Part of the reason why the DMV performs so well on WSJ5 and is still competitive on WSJ10 and WSJ15 is that the data itself precludes it from realizing the very fertile heads that it seeks. As sentence length increases, the DMV gets increasingly lost, excited by the possibilities, while other models could do better.

7 Replacing the Ad-Hoc Harmonic

We tried out a cute idea of a more principled, yet still unsupervised approach to initializing $\mathbb{P}_{\texttt{ATTACH}}$: clustering using an SVM! Essentially, some attachments are likely; others are not — thus, we could pretend that there are exactly two classes. Furthermore, for every sentence of length n, of the $\Theta(n^2)$ possible links, exactly $n-1$ will be correct — thus, we even know the relative sizes of these clusters! We encode the same information given to the ad-hoc harmonic (the head class, the attachment class, direction and distance) using binary-valued features and dump it all on SVM^{light} without labeling a single example, but stating that the prior for the positive class is, in the case of WSJ10, $\frac{44,826}{362,952} \approx 0.1235$, using its transduction option `-p` and the basic linear kernel:

35.194%	(53.434%)	4.0420	[24.593%	(43.895%)	3.3409]	[Ad-Hoc Harmonic Seed Round]
36.600%	(53.164%)	4.1302	[24.360%	(42.960%)	3.3932]	[SVMonic Seed Round]
53.839%	(65.492%)	2.6880	[53.391%	(64.843%)	2.6346]	[Ad-Hoc Harmonic Viterbi Result]
49.669%	(62.990%)	2.7104	[50.172%	(62.893%)	2.6525]	[SVMonic Viterbi Result]

The numbers are in the same ball-park — presumably we could augment our features with more rich information, e.g. actual intervening word classes, as well as play with more sophisticated kernel functions.

8 Testing Independence Assumptions

From the point of view of statistical hypothesis testing, pretty much all relations gathered by the supervised oracle appear significantly non-independnet, even when divergence is tiny. And here, abundance of data back-fires — as more data is gathered, it provides extra evidence against the independence of less and less dependent events. As a thought experiment, inspired by the *Kolmogorov-Smirnov Test*, we introduce the notion of *external misallocation* — one minus the total probability assigned to jointly observed events, with each "independent" factor estimated by an MLE, i.e. the total probability mass left over for unseen events. Given that we aren't explicitly aiming to smooth, we hope for this number to be small. As it turns out, for some parts of speech, this number is empirically as high as 30-50%, in particular for many verb types and the coordinating conjunction! We also explored *internal misallocation* — the total probability mass assigned to observed events in excess of their actual frequency. For certain types of verbs, these numbers too can be high, at 10-15%.

9 A Joint Model

Based on several pieces of insights already gathered, we introduce the jDMV, which attempts to model as much as possible jointly, while preserving much of the elegance of the DMV. In particular, note that of n words in a sentence, only $n-1$ become children — a priori, the expected number of children is less than one! Fertile productions are rare, and we term those with at most one child per side *compact*. Also, recurion complicates things (recall that adjacency trumped recursive non-adjacency), so we try to do as much as possible in the infancy, while the head is small and still knows everything, rather than when it matures and its world-view is clouded by recursion, having lost track of what exactly went into making it the way it finds itself. Also, since the problem is constrained, it could be that taking care of the small nodes would help the bigger ones take care of themselves...

For the jDMV, we fuse attachment probabilities with non-adjacent stopping, yielding two probability distributions to be estimated: $\mathbb{P}_{\texttt{INIT}}(c_h, c_l, c_r, s_l, s_r)$ and $\mathbb{P}_{\texttt{RECURSE}}(c_h, c_a, s, dir)$, where the initial probability reflects the chances that c_h attaches c_l to its left, c_r to its right, and then stops, in either direction, based on $s_l, s_r \in \{\texttt{T}, \texttt{F}\}$; recursive probability reflects the

chances that it goes on to attaches c_a on side $dir \in \{\texttt{L}, \texttt{R}\}$, and then stops, based on s, given that it already has at least one child on that side. This model could be augmented to condition on the number of children as well, but we wanted to derive complete and efficient EM/IO re-estimation, in addition to the Viterbi approximation. Unfortunately, we are running out of time and space to present these results...

10 A Hint of Impossibility, a Glimpse of Hope, and a Wealth of Data...

Pereira and Schabes showed that pure, unsupervised EM fails to infer even the small artificial grammar

$$\begin{array}{l} C \rightarrow SA; \\ D \rightarrow SB; \\ A \rightarrow a; \\ B \rightarrow b; \end{array} \qquad S \rightarrow \begin{cases} AC, \text{ w.p. } 0.4; \\ BD, \text{ w.p. } 0.4; \\ AA, \text{ w.p. } 0.1; \\ BB, \text{ w.p. } 0.1. \end{cases}$$

generating the two-symbol palindrome language $L = \{ww^R \mid w \in \{a, b\}^+\}$, but manages to recover if given some partial bracketing constraints, which enabled 1) faster chart generation; 2) faster convergence; and 3) convergence to the true answer.

One of our hopes was that mark-up (e.g. hyper-linked anchor text on the web, fonts like bold and italics, etc.) may frequently delimit linguistically sound constituents. If so, then there could be a (free, growing and multi-lingual) wealth of partial bracketing data to be mined on the web. To this end, we down-loaded one news-style blog in its entirety, hacked around `MxTerminator` to break clean, pretty-printed HTML into sentences, then used *Charniak's Parser* for (supervised) POS-tagging and parsing, a la the PTB. Excluding (useless) full-sentence and single-word annotations, we were able to extract 4,826 bracketings from a total of 55,012 sentences (that's 8.77% yield); 37.2% of these matched a constituent produced by the parser perfectly (presumably others could still be helpful, since we are aiming for dependency and not constituency parsing); 70.6% (vast majority) of the extracted bracketings are derived by noun phrases, while 10.1% (a non-trivial minority) fall under verb phrases. All of this suggests that we may be able to perform some very interesting lightly supervised learning using these partial bracketings as valuable hints to the otherwise unsupervised algorithms!

Well, we did this, but there is no room to include these results either. `:(`

11 References Consulted

Collins, M. Head-Driven Statistical Models for Natural Language Parsing. 1999.
Ph.D. Thesis, MIT.

Collins, M. Three Generative, Lexicalised Models for Statistical Parsing. In *ACL 35*, 16-23.

Charniak, E.
`ftp://ftp.cs.brown.edu/pub/nlparser/parser05Aug16.tar.gz`

Hardt, D.
`http://www.id.cbs.dk/~dh/corpus/tools/MXTERMINATOR.html`

Jurafsky, D., and J.H.Martin. Speech and Language Processing:
An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.
2008. 2nd ed.

Klein, D. The Unsupervised Learning of Natural Language Structure. 2005.
Ph.D. Thesis, Stanford University.

Klein, D., and C.D.Manning. Corpus-Based Induction of Syntactic Structure:
Models of Dependency and Constituency. In *ACL 42*, 479-486.

McDonald, R. and J.Nivre. Introduction to Data Driven Dependency Parsing.
`http://dp.esslli07.googlepages.com/`

Pereira, F., and Y.Schabes. Inside-outside reestimation from partially bracketed corpora. In *ACL 30*, 128-35.

Xia, F. Inner loop of the Inside-outside algorithm.
`http://courses.washington.edu/ling570/fei_fall07/12_5_Summary.pdf`

Xia, F. Inside-outside algorithm.
`http://faculty.washington.edu/fxia/courses/LING572/inside-outside.ppt`