Statistical and Learning Methods in Natural Language Processing

Ido Dagan   Shuly Wintner

Spring 2004

Shallow parsing

- **Shallow parsing** consists of identifying the main components of sentences and their heads and determining syntactic relationships among them.
- Problem: Given an input string $O = \langle o_1, \ldots, o_n \rangle$, a **phrase** is a consecutive substring $\langle o_i, \ldots, o_j \rangle$. The goal is, given a sentence, to identify all the phrases in the string. A secondary goal is to tag the phrases as Noun Phrase, Verb Phrase etc. An additional goal is to identify relations between phrases, such as subject–verb, verb–object etc.
- Question: How can this problem be cast as a classification problem?

Shallow parsing

- Lance A. Ramshaw and Mitchell P. Marcus, **Text chunking using transformation-based learning**, in *Proceedings of the Third Annual Workshop on Very Large Corpora*, 1995.
- Walter Daelemans, Sabine Buchholz and Jorn Veenstra, **Memory-based shallow parsing**, in *Proceedings of CoNLL*, 1999.
- Vasin Punyakanok and Dan Roth, **The use of classifiers in sequential inference**, in *Proceedings of NIPS'00*, 2000.

Text chunking

- Lance A. Ramshaw and Mitchell P. Marcus, **Text chunking using transformation-based learning**, in *Proceedings of the Third Annual Workshop on Very Large Corpora*, 1995.
- Text chunking involves dividing sentences into non-overlapping segments on the basis of fairly simple superficial analysis.
- This is a useful and relatively tractable precursor to full parsing, since it provides a foundation for further levels of analysis, while still allowing complex attachment decisions to be postponed to a later phase.

Deriving chunks from treebank parses

- Annotation of training data can be done automatically based on the parsed data of the Penn Tree Bank
- Two different chunk structure tagsets: one bracketing non-recursive "base NPs", and one which partitions sentences into non-overlapping N-type and V-type chunks

"Base NP" chunk structure

- The goal of the "base NP" chunks is to identify essentially the initial portions of non-recursive noun phrases up to the head, including determiners but not including postmodifying prepositional phrases or clauses.
- These chunks are extracted from the Treebank parses, basically by selecting NPs that contain no nested NPs.
- The handling of conjunction follows that of the Treebank annotators as to whether to show separate baseNPs or a single baseNP spanning the conjunction.
- Possessives are treated as a special case, viewing the possessive marker as the first word of a new baseNP, thus flattening the recursive structure in a useful way.

"Base NP" chunk structure

Example

$[_N$ The government $_N]$ has $[_N$ other agencies and instruments $_N]$ for pursuing $[_N$ these other objectives $_N]$.

"Base NP" chunk structure

Example

Even [$_N$ Mao Tse-tung $_N$] [$_N$'s China $_N$] began in [$_N$ 1949 $_N$] with [$_N$ a partnership $_N$] between [$_N$ the communists $_N$] and [$_N$ a number $_N$] of [$_N$smaller , non-communist parties $_N$] .

Partitioning chunks

- In the partitioning chunk experiments, the prepositions in prepositional phrases are included with the object NP up to the head in a single N-type chunk.
- The handling of conjunction again follows the Treebank parse.
- The portions of the text not involved in N-type chunks are grouped as chunks termed V-type, though these "V" chunks include many elements that are not verbal, including adjective phrases.
- Again, the possessive marker is viewed as initiating a new N-type chunk.

Partitioning chunks

Example

$[_N$ Some bankers $_N]$ $[_V$ are reporting $_V]$ $[_N$ more inquiries than usual $_N]$ $[_N$ about CDs $_N]$ $[_N$ since Friday $_N]$.

Partitioning chunks

Example

$[_N$ Indexing $_N]$ $[_N$ for the most part $_N]$ $[_V$ has involved simply buying $_V]$ $[_V$ and then holding $_V]$ $[_N$ stocks $_N]$ $[_N$ in the correct mix $_N]$ $[_V$ to mirror $_V]$ $[_N$ a stock market barometer $_N]$.

Encoding chunking as a tagging problem

- Each word carries both a part-of-speech tag and also a “chunk tag” from which the chunk structure can be derived.
- In the baseNP experiments, the chunk tag set is $\{I, O, B\}$, where words marked I are *inside* some baseNP, those marked O are *outside*, and the B tag is used to mark the leftmost item of a baseNP which immediately follows another baseNP.
- In the partitioning experiments, the chunk tag set is $\{BN, N, BV, V, P\}$, where BN marks the first word and N the succeeding words in an N-type group while BV and V play the same role for V-type groups.

Encoding chunking as a tagging problem

- Encoding chunk structure with tags attached to words (rather than inserting bracket markers between words) limits the dependence between different elements of the encoded representation.
- While brackets must be correctly paired in order to derive a chunk structure, it is easy to define a mapping that can produce a valid chunk structure from any sequence of chunk tags; the few hard cases that arise can be handled locally.
- For example, in the baseNP tag set, whenever a B tag immediately follows an O, it must be treated as an I.
- In the partitioning chunk tag set, wherever a V tag immediately follows an N tag without any intervening BV, it must be treated as a BV.

Baseline

- Transformational learning begins with some initial "baseline" prediction, which here means a baseline assignment of chunk tags to words.
- Reasonable suggestions for baseline heuristics after a text has been tagged for part-of-speech might include assigning to each word the chunk tag that it carried most frequently in the training set, or assigning each part-of-speech tag the chunk tag that was most frequently associated with that part-of-speech tag in the training.
- Testing both approaches, the baseline heuristic using part-of-speech tags turned out to do better.

Rule templates

- Rules can refer to words and to POS tags. Up to three words to the left and right of the target word, and up to two POS tags to the left and right of the target can be addressed.
- A set of 100 rule templates, obtained by the cross product of 20 word-patterns and 5 tag-patterns, was used.
- Then, a variant of Brill's TBL algorithm was implemented.

Results

- BaseNP chunks:

Training	Recall	Precision
Baseline	81.9%	78.2%
50K	90.4%	89.8%
100K	91.8%	91.3%
200K	92.3%	91.8%

- Partitioning chunks:

Training	Recall	Precision
Baseline	60.0%	47.8%
50K	86.6%	85.8%
100K	88.2%	87.4%
200K	88.5%	87.7%

Memory-based shallow parsing

- Walter Daelemans, Sabine Buchholz and Jorn Veenstra, **Memory-based shallow parsing**, in *Proceedings of CoNLL*, 1999.
- Shallow parsing consists of discovering the main constituents of sentences (NPs, VPs, PPs) and their heads, and determining syntactic relationships (like subjects, objects or adjuncts) between verbs and heads of other constituents.
- This is an important component of text analysis systems in applications such as information extraction and summary generation.

Memory-based learning: reminder

- A memory-based learning algorithm constructs a classifier for a task by storing a set of examples.
- Each example associates a feature vector (the problem description) with one of a finite number of classes (the solution).
- Given a new feature vector, the classifier extrapolates its class from those of the most similar feature vectors in memory.
- The metric defining similarity can be automatically adapted to the task at hand.

Organization

- Syntactic analysis is carved up into a number of classification tasks.
- These can be segementation tasks (e.g., deciding whether a word or tag is the beginning or the end of an NP) or disambiguation tasks (e.g., deciding whether a chunk is the subject, object or neither).
- Output of one module (e.g., POS tagging or chunking) is used as input by other modules (e.g., syntactic relation assignment).

Algorithms and implementation

- All the experiments use TiMBL.
- Two variants of MBL are used:

  IB1-IG: The distance between a test item and a memory item is the number of features on which they disagree. The algorithm uses information gain to weigh the cost of mismatches. Classification speed is linear in the number of training instances times the number of features.

  IGTREE: A decision tree is created with features as tests, ordered according to information gain of features. Classification speed is linear in the number of features times the average branching factor of the tree, which is bound by the average number of values per feature.

Experiments

- Two series of experiments:
  - Memory-based NP and VP chunking
  - Subject/object detection using the chunker

Chunking as a tagging task

- Each word is assigned a tag which indicates whether it is inside or outside a chunk:

I_NP	inside a baseNP
O	outside both a baseNP and a baseVP
B_NP	inside a baseNP, but the preceding word is in another baseNP
I_VP	inside a baseVP
B_VP	inside a baseVP, but the preceding word is in another baseVP

- Since baseNPs and baseVPs are non-overlapping and non-recursive, these five tags suffice to unambiguously chunk a sentence.

Tagging example

Example

[$_{NP}$ Pierre$_{I_NP}$ Vinken$_{I_NP}$ $_{NP}$] ,$_{O}$ [$_{NP}$ 61$_{I_NP}$ years$_{I_NP}$ $_{NP}$] old$_{O}$,$_{O}$ [$_{VP}$ will$_{I_VP}$ join$_{I_VP}$ $_{VP}$] [$_{NP}$ the$_{I_NP}$ board$_{I_NP}$ $_{NP}$] as$_{O}$ [$_{NP}$ a$_{I_NP}$ nonexecutive$_{I_NP}$ director$_{I_NP}$ $_{NP}$] [$_{NP}$ Nov.$_{B\_NP}$ 29$_{I_NP}$ $_{NP}$] .

Chunking as a tagging task: experiments

- The features for the experiments are the word and the POS tag (as provided by the Penn Tree Bank) of two words to the left, the target word and one word to the right.
- The baseline is computed with IB1-IG, using as features only the focus word/POS.

Results

- BaseNP chunks:

Method	Recall	Precision
Baseline words	79.7%	76.2%
Baseline POS	82.4%	79.5%
IGTree	93.1%	91.8%
IB1-IG	94.0%	93.7%

- BaseVP chunks:

Method	Recall	Precision
Baseline words	73.4%	67.5%
Baseline POS	87.7%	74.7%
IGTree	94.2%	93.0%
IB1-IG	95.5%	94.0%

Subject/object detection

- Finding the subject or object of a verb is defined as a mapping from pairs of words (the verb and the head of the constituent), and a representation of their context, to a class (subject, object or neither).
- A verb can have multiple subjects (in the case of NP coordination) and a word can be the subject of more than one verb (VP coordination).
- The input is POS tagged and chunked.

  Example

  $[_{NP}$ My/PRP sisters/NNS $_{NP}]$ $[_{VP}$ have/VBP not/RB seen/VBN $_{VP}]$ $[_{NP}$ the/DT old/JJ man/NN $_{NP}]$ lately/RB ./.

- All chunks are reduced to their heads, defined as the rightmost word of a baseNP or baseVP.

Subject/object detection: features

- The distance, *in chunks*, between the verb and the head
- The number of other baseVPs between the verb and the head
- The number of commas between the verb and the head
- The verb and its POS tag
- The head and its POS tag
- The two left context and one right context words/chunks of the head, represented by the word and its POS tag

Subject/object detection: results

Finding unrestricted subjects and objects is hard:

Method	Together	Subjects	Objects
Heuristic baseline	66.2	65.2	67.7
IGTree	76.2	75.8	76.8
IB1-IG	75.6	76.5	74.0
Unanimous	77.8	77.1	79.0

The use of classifiers in sequential inference

- Vasin Punyakanok and Dan Roth, **The use of classifiers in sequential inference**, in *Proceedings of NIPS'00*, 2000.
- Combination of the outcome of several classifiers in a way that provides a coherent inference that satisfies some constraints.
- Two general approaches to identifying phrase structure: *projection-based Markov models* and *constraint satisfaction with classifiers*.

Identifying phrase structure

- Classifiers recognize in the input string local signals which are indicative of the existence of phrases
- Classifiers can indicate that an input symbol is **i**nside or **o**utside a string or that a symbol **o**pens or **c**loses a string.
- The open/close approach has been found more robust and is pursued here.
- The classifiers' outcomes can be combined to determine the phrase, but this combination must satisfy certain constraints for the result to be legitimate.
- Several types of constraints, such as length, order and others, can be formalized and incorporated into the two approaches studied here.

Identifying phrase structure

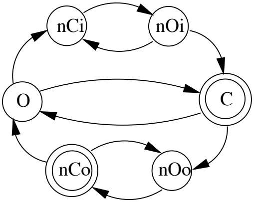
- Two complex phrase identification tasks are defined: base NPs and Subject-Verb patterns.

  Example
  [The theory presented claims] that [the algorithm runs] and performs ...

- Two classifiers are learned for each task, predicting whether word *t* **o**pens or **c**loses a phrase.

Identifying phrase structure

- Each classifier may output two values: open/¬open and close/¬close.
- However, for technical reasons, *three* values are output by each classifier, where the 'not' value is divided according to whether or not the word is inside a phrase.
- Consequently, the values are: **O, nOi, nOo, C, nCi, nCo**.
- The order of these values is constrained according to the following diagram:



Definitions

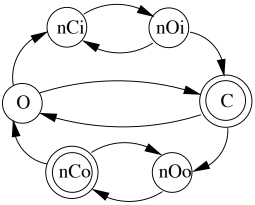
- The input string is $O = \langle o_1, o_2, \ldots, o_n \rangle$
- A phrase $\pi^{i,j}(O)$ is a substring $\langle o_1, o_{i+1}, \ldots, o_j \rangle$ of O
- $\pi^*(O)$ is the set of all possible phrases of O
- $\pi^{i,j}(O)$ and $\pi^{k,l}(O)$ **overlap**, denoted $\pi^{i,j}(O) \rightleftharpoons \pi^{k,l}(O)$, iff $j \geq k$ and $l \geq i$
- Given a string O and a set Y of classes of phrases, a solution to the phrase identification problem is a set $\{(\pi, y) \mid \pi \in \pi^*(O) \text{ and } y \in Y\}$ such that for all $(\pi_i, y), (\pi_j, y)$, if $i \neq j$ then $\pi_i \not\rightleftharpoons \pi_j$
- We assume that $|Y| = 1$.

Hidden Markov Model combinator

Reminder: an HMM is a probabilistic finite state automaton consisting of

- A finite set S of states
- A set O of observations
- An initial state distribution $P_1(s)$
- A state-transition distribution $P(s|s')$ for $s, s' \in S$, and
- An observation distribution $P(o|s)$ for $o \in O, s \in S$.
- Constraints can be incorporated into the HMM by constraining the state transition probability distribution. For example, set $P(s|s') = 0$ for cases where the transition from s' to s is not allowed.

Hidden Markov Model combinator

- We assume that we have local signals which indicate the state. That is, classifiers are given with states as their outcome.
- Formally, we assume that $P_t(s|o_y)$ is given, where t is a time step in the sequence.
- 

- Constraints on state transitions do not have to be stated explicitly; they can be recovered from training data.

Hidden Markov Model combinator

- Instead of estimating the observation probability $P(o|s)$ directly from training data, it is computed from the classifiers' output:
$$P_t(o_t|s) = \frac{P_t(s|o_t) \times P_t(o_t)}{P_t(s)}$$
- $$P_t(s) = \Sigma_{s' \in S} P(s|s') \times P_{t-1}(s')$$
where $P_1(s)$ and $P(s|s')$ are the standard HMM distributions
- $P_t(o_t)$ can be treated as constant since the observation sequence is fixed for all compared sequences.
- The Viterbi algorithm can be used to find the most likely state sequence for a given observation.

Projection based Markov Model combinator

- In standard HMMs, observations are allowed to depend only on the current state; no long-term dependencies can be modeled.
- Similarly, constraint structure is restricted by having a stationary probability distribution of a state given the previous state.
- In PMM, these limitations are relaxed by allowing the distribution of a state to depend, in addition to the previous state, on the observation.
- Formally, the independence assumption is:

$$P(s_t | s_{t-1}, \ldots, s_1, o_{t-1}, \ldots, o_1) = P(s_t | s_{t-1}, o_t)$$

Projection based Markov Model combinator

- Given an observation sequence O, the most likely state sequence S given O is obtained by maximizing

$$
\begin{aligned}
P(S|O) &= \Pi_{t=2}^{n}[P(s_t|s_1,\ldots,s_{t-1},o)]P_1(s_1|o) \\
&= \Pi_{t=2}^{n}[P(s_t|s_{t-1},o_t)]P_1(s_1|o_1)
\end{aligned}
$$

- In this model the classifiers decisions are incorporated in the terms $P(s|s',o)$ and $P_1(s|o)$. The classifiers take into account not only the current input symbol but also the previous state. The hope is that these new classifiers perform better because they are given more information.

Constraint satisfaction based combinator

- A boolean constraint satisfaction problem consists of a set of n variables $V = \{v_1, \ldots, v_n\}$, each ranging over values in a domain D_i (here, $0/1$).
- A constraint is a relation over a subset of the variables, defining a set of "global" possible assignments to the referred variables.
- A solution to a CSP is an assignment that satisfies all the constraints.
- The CSP formalism is extended to deal with probabilistic variables; the solution now has to minimize some cost function. Thus, each variable is associated with a cost function $c_i : D_i \rightarrow \Re$.

Constraint satisfaction with classifiers

- Given an input $O = \langle o_1, \ldots, o_l \rangle$, let $V = \{v_{i,j} \mid 1 \leq i \leq j \leq l\}$. Each variable $v_{i,j}$ corresponds to a potential phrase $\pi^{i,j}(O)$.
- Associate with each variable $v_{i,j}$ a cost function $c_{i,j}$.
- Constraints can now be expressed as boolean formulae. For example, the constraint that requires that no two phrases overlap is expressed as:

$$\bigwedge_{\pi^{a,b} \rightleftharpoons \pi^{c,d}} (\neg v_{a,b} \vee \neg v_{c,d})$$

- The solution is an assignment of 0/1 to variables which satisfies the constrains and, in addition, minimizes the overall cost

$$\Sigma_{i=1}^{n} c_i(v_i)$$

Constraint satisfaction with classifiers

- In general, the corresponding optimization problem is NP-hard.
- In the special case where costs are in $[0, 1]$, a solution which is at most twice the optimal can be found efficiently.
- For the specific case of non-overlapping phrase identification, the problem can be solved efficiently using a graph representation of the constraints (the problem reduces to finding a shortest path in a weighted graph).
- What is left is determining the cost function.

Constraint satisfaction with classifiers: cost function

- It can be shown that in order to maximize the number of correct phrases, each phrase has to be assigned a cost that is minus the probability of the phrase being correct:

$$c_{i,j}(v_{i,j}) = \begin{cases} -p_{i,j} & \text{if } v_{i,j} = 1 \\ 0 & \text{otherwise} \end{cases}$$

where $p_{i,j}$ is the probability that the phrase $\pi^{i,j}$ is correct.

Constraint satisfaction with classifiers: cost function

- Assuming independence between symbols in a phrase, and assuming that the important part of a phrase are only its beginning and end words,

$$p_{i,j} = P_i^O(O) \times P_j^C(C)$$

  where $P_i^O(O)$ is the probability that the first symbol o_i in the phrase is actually the beginning of a phrase, and $P_j^C(C)$ is the probability that the last symbol o_j of the phrase is actually the end of a phrase.

- These two probabilities are supplied by the classifiers.

Results

	NP		SV	
Method	POS only	POS + word	POS only	POS + word
HMM	90.64	92.89	64.15	77.54
PMM	90.61	92.98	74.98	86.07
CSCL	90.87	92.88	85.36	90.09