Toward Better Understanding of Hebrew NP Chunks

Yoav Goldberg and Michael Elhadad
Computer Science Department
Ben Gurion University of the Negev
P.O.B 653 Be'er Sheva 84105, Israel
yoavg,elhadad@cs.bgu.ac.il

Abstract

In (Goldberg and Elhadad, 2007) we presented two techniques (SVM Model Tampering and Anchored Learning) for investigating the SVM learning process and resulting models. These techniques were applied to the task of SVM based Hebrew NP Chunking. The results were better understanding of SVM based chunking, of the role lexical features play in the learned models, of the Hebrew NP chunking task definition, and identification of several deficencies in the NP Chunking corpus. This paper focuses on the Hebrew specific issues raised by that work, which are presented with more detail.

1 Introduction and Previous Work

NP chunking is the task of marking the boundaries of simple noun-phrases in text. It is a well studied problem in English, and was the focus of CoNLL2000's Shared Task (Sang and Buchholz, 2000). NP chunks in the shared task data are Base NPs, which are non-recursive NPs. This definition yields good NP chunks for English as well as other languages. Indeed, non-recursiveness is a key component in the definition of text chunks (see (Abney, 1991; Abney, 1996; Marcus and Ramshaw, 1995) for English, and (Schiehlen, 2002; Federici et al., 1998; Ola et al., 2003; Zhang and Zhou, 2002) for German, Italian, Swedish and Chinease respectively).

In (Goldberg et al., 2006) , we argued that the non-recursiveness criterion is not applicable to Hebrew NP Chunks, mainly because of the prevalence of the Hebrew construct state ($smixut$), which appears in about 40% of Hebrew NPs. $Smixut$ is similar to a noun-compound construct, but one that can join a noun (with a special morphological marking) with a full NP.

We proposed an alternative definition (termed **SimpleNP**) for Hebrew NP chunks. A SimpleNP cannot contain embedded relatives, prepositions, VPs and NP-conjunctions (except when they are licensed by $smixut$). It can contain $smixut$, possessives (even when they are attached by the 'של/of' preposition) and partitives (and, therefore, allows for a limited amount of recursion). As we will see in section 4.3, this definition is challenged by empirical data. Section 5 also presents a discussion of the problems involved in defining Hebrew SimpleNPs, as well as a proposed refined definition.

Following Ramshaw and Marcus (1995), the current dominant approach for text chunking is to formulate chunking as a classification (or tagging) task, in which each word is classified as the (B)eginning, (I)nside or (O)outside of a chunk. Features for this classification usually involve local context features. Kudo and Matsumoto (2000) used SVM as a classification engine and achieved an F-Score of 93.79 on the English shared task data.

(Goldberg et al., 2006) established that while some chunking methods that perform well for English (Marcus and Ramshaw, 1995; Cardie and Pierce, 1998) do not transfer well to Hebrew SimpleNPs, the SVM-based chunking of (Kudo and Matsumoto, 2000) works well also for Hebrew SimpleNPs. It was further shown that the use of two Hebrew-specific morphological features, namely

"number" and "construct-state" improve chunking results, and that the effect of these features was magnified when dealing with noisy POS tags. However, most of the success was attributed to SVM's ability to use lexical features. Indeed, incorporating lexical knowledge into the model improved chunking F-Score from 78 to over 92. Section 3 refines this conclusion by showing that only a few closed-class lexical features are responsible for most of the improvement. The relative importance of specific lexical features is discussed.

Finally, the construction of the NP Chunks Corpus in (Goldberg et al., 2006) was achieved by use of a heuristic for automatically extracting Simple NPs from the Hebrew Treebank (Sima'an et al., 2001). Because the tagging in the Treebank is slightly inconsistent and the heuristic was not fully verified, this approach resulted in a highly unreliable chunking corpus. (Goldberg and Elhadad, 2007) describes Anchored Learning, a technique for identifying training instances (corpus locations) that are "hard for the (SVM) classifiers to learn." These locations are either corpus errors, or genuinely hard cases. Systematic corpus errors that were found using the Anchored Learning analysis are discussed in Section 4.2. Some of the genuinely hard cases and their implications are discussed in Section 4.4. We expect in future work to refine the notion of NP chunk in Hebrew on the basis of the data gathered.

2 Technical Background

Our approach to NP chunking is based on SVM classification. This section describes SVM-based chunking, as well as the Model Tampering and Anchored Learning techniques. These techniques allow us to understand the nature of the models learned by the SVM and refine the feature set we use to describe the data.

2.1 SVM Based Chunking

SVM (Vapnik, 1995) is a supervised binary classifier. The input to the learner is a set l training samples $(x_1, y_1), \ldots, (x_l, y_l)$, $x \in R^n$, $y \in \{+1, -1\}$. x_i is an n dimensional feature vector representing the ith sample, and y_i is the label for that sample. The result of the learning process is the set SV of Support Vectors, the associated weights α_i, and a constant b. The Support Vectors are a subset of the training vectors, and together with the weights and b they define a hyperplane that optimally separates the training samples. The basic SVM formulation is of a linear classifier, but by introducing a kernel function K that non-linearly transforms the data from R^n into a higher dimensional space, SVM can be used to perform non-linear classification. SVM's decision function is: $y(x) = sgn\left(\sum_{j \in SV} y_j \alpha_j K(x_j, x) + b\right)$ where x is an n dimensional feature vector to be classified. In the linear case, K is a dot product operation and the sum $w = \sum y_j \alpha_j x_j$ is an n dimensional weight vector assigning weight for each of the n features. The other kernel function we consider in this paper is a polynomial kernel of degree 2: $K(x_i, x_j) = (x_i \cdot x_j + 1)^2$. When using binary valued features, this kernel function essentially implies that the classifier considers not only the explicitly specified features, but also all available pairs of features. In order to cope with inseparable data, the learning process of SVM allows for some misclassification, the amount of which is determined by a parameter C, which can be thought of as a penalty for each misclassified training sample.

In SVM based chunking, each word and its context is considered a learning sample. We refer to the word being classified as w_0, and to its part-of-speech (PoS) tag, morphology, and B/I/O tag as p_0, m_0 and t_0 respectively. The information considered for classification is $w_{-cw} \ldots w_{cw}$, $p_{-cp} \ldots p_{cp}$, $m_{-cm} \ldots m_{cm}$ and $t_{-ct} \ldots t_{-1}$. The feature vector F is an indexed list of all the features present in the corpus. A feature f_i of the form $w_{+1} = dog$ means that the word following the one being classified is 'dog'. Every learning sample is represented by an $n = |F|$ dimensional binary vector x. $x_i = 1$ iff the feature f_i is active in the given sample, and 0 otherwise. This encoding leads to extremely high dimensional vectors, due to the lexical features $w_{-cw} \ldots w_{cw}$.

In the classification view to text chunking, each word is classified into one of 3 classes: {B,I,O}, while SVM is a binary classifier. In order to use SVM for text chunking, 3 different classifiers ({B/O,I/O,B/I}[1]) are trained and their majority vote

[1] One can also train a different set of classifiers:

is taken.

2.2 Model Tampering and Anchored Learning

An important observation about SVM classifiers is that features which are not active in any of the Support Vectors have no effect on the classifier decision.

Model Tampering is a procedure in which we change the Support Vectors in a model by forcing some values in the vectors to 0.

The result of this procedure is a new Model in which the deleted features never take part in the classification.

By observing the performance of these new Models, we can learn about the importance of these deleted features on the classification.

Anchored Learning is a procedure in which a unique feature a_i (an *anchor*) is added to each training sample (we add as many new features to the model as there are training samples). These new features make the data linearly separable. Intuitively, because the anchor features have no generalization ability, they are given very little "weights" in the resulting model. However, samples that are hard to learn on their own can now be easily "memorized" by the SVM learner using their anchor feature. Thus, anchor features with high weights in the SVM classification model are indication of hard-to-learn samples. These samples may correspond to mere errors in the corpus, or genuinely hard-to-decide cases.

3 The Role of Lexical Features

Goldberg *et al.* (2006) have established that using lexical features increases the chunking F-measure from 78 to over 92 on the Hebrew Treebank. We refine this observation by using Model Tampering, in order to assess the importance of lexical features in NP Chunking. We are interested in identifying which specific lexical items and contexts impact the chunking decision, and quantifying their effect. Our method is to train a chunking model on a given training corpus, tamper with the resulting model in various ways and measure the performance [2] of the tampered models on a test corpus.

{B/NonB,I/NonI,O/NonO}. The results are similar, but the classifirs in the set used in this article are faster to train.

[2]The performance metric we use is the standard Precision/Recall/F measures, as computed by the conlleval program: http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt

3.1 Experimental Setting

We use the corpora of (Goldberg et al., 2006). The first one is derived from the original Treebank by projecting the full syntactic tree, constructed manually, onto a set of NP chunks according to the SimpleNP rules. We refer to the resulting corpus as HEB_{Gold} since PoS tags are fully reliable. The HEB_{Err} version of the corpus is obtained by projecting the chunk boundaries on the sequence of PoS and morphology tags obtained by the automatic PoS tagger of Adler & Elhadad (2006). This corpus includes an error rate of about 8% on PoS tags. The first 500 sentences are used for testing, and the rest for training. The corpus contains 27K NP chunks. Training was done using Kudo's YAMCHA toolkit[3]. The models were trained using a polynomial kernel of degree 2, with $C = 1$. The features used were $w_{-2} \dots w_2$, $p_{-2} \dots p_2$, $t_{-2} \dots t_{-1}$ and $m_{-2} \dots m_2$ (these are the same settings as in (Goldberg et al., 2006)).

3.2 Tamperings

We experimented with the following tamperings:

TopN – We define *model feature count* to be the number of Support Vectors in which a feature is active in a given classifier. This tampering leaves in the model only the top N lexical features in each classifier, according to their count.

NoPOS – all the lexical features corresponding to a given part-of-speech are removed from the model. For example, in a NoJJ tampering, all the features of the form $w_i = X$ are removed from all the support vectors in which $p_i = JJ$ is active.

Loc≠i – all the lexical features with index i are removed from the model *e.g.*, in a Loc≠+2 tampering, features of the form $w_{+2} = X$ are removed).

Loc=i – all the lexical features with an index other than i are removed from the model.

3.3 Results and Discussion

Highlights of the results are presented in Tables (1-3). The numbers reported are F measures.

The results of the TopN tamperings show that most of the lexical features are irrelevant for the classification – the numbers achieved by using all the lexical features (about 30,000) are very close

[3]http://chasen.org/∼taku/software/yamcha/

TopN	HEB_{Gold}	HEB_{Err}
ALL	93.58	92.48
N=0	78.32	76.27
N=10	90.21	88.68
N=50	91.78	90.85
N=100	92.25	91.62
N=500	93.60	92.23
N=1000	93.56	92.41

Table 1: Results of TopN Tampering.

to those obtained using only a few lexical features. This finding is very encouraging, and suggests that SVM based chunking is robust to corpus variations.

Another conclusion is that lexical features help balance the fact that PoS tags can be noisy: HEB_{Err} include about 8% PoS tagging errors. While in the case of "perfect" PoS tagging (HEB_{Gold}), a very small amount of lexical features is sufficient to reach the best F-result (500 out of 30,264), in the presence of PoS errors, more than the top 1000 lexical features are needed to reach the result obtained with all lexical features.

More striking is the fact that in the top 10 lexical features are responsible for an improvement of 12.4 in F-score. The words covered by these 10 features include: `Start of Sentence marker` and `comma`, `quote`, '`of/`של', '`and/`ו', '`the/`ה' and '`in/`ב'.

This finding suggests that the Hebrew PoS tagset might not be informative enough for the chunking task, especially where punctuation [4] and prepositions are concerned.

It appears that the של preposition behave diffrently than the other prepositions. Indeed, in the English Penn Tagset, possesors are distinguished from regular prepositions by means of a special tag (POS).

The results in Table 2 deepens our understanding of the relative relevance of different lexical features.

NoPOS	HEB_G	HEB_E	NoPOS	HEB_G	HEB_E
Prep	**85.25**	**84.40**	Pronoun	92.97	92.14
Punct	**88.90**	**87.66**	Conjunction	92.31	91.67
Adverb	92.02	90.72	Determiner	92.55	91.39

Table 2: Results of Hebrew NoPOS Tampering. Other scores are $\geq 93.3(HEB_G)$, $\geq 92.2(HEB_E)$.

[4] Unlike the WSJ PoS tagset in which most punctuations get unique tags, our tagset treat punctuation marks as one group.

When removing lexical features of a specific PoS, the most dramatic loss of F-score is reached for Prepositions and Punctuation marks, followed by Adverbs, and Conjunctions. Strikingly, lexical information for most open-class PoS (including Proper Names and Nouns) has very little impact on Hebrew chunking performance.

From these observations, one could conclude that enriching a model based only on PoS with lexical features for only a few closed-class PoS (prepositions and punctuation) could provide appropriate results even with a simpler learning method, one that cannot deal with a large number of features. We tested this hypothesis by training the Error-Driven Pruning (EDP) method of (Cardie and Pierce, 1998) with an extended set of features. EDP with PoS features only produced an F-result of 76.3 on HEB_{Gold}. By adding lexical features only for prepositions {של כ ה ב מ}, one conjunction {ו} and punctuation, the F-score on HEB_{Gold} indeed jumps to 85.4. However, when applied on HEB_{Err}, EDP falls down again to 59.4. This striking disparity, by comparison, lets us appreciate the resilience of the SVM model to PoS tagging errors, and its generalization capability even with a reduced number of lexical features.

Another implication of this data is that commas and quotation marks play a major role in determining NP boundaries in Hebrew. Goldberg *et al.* (2006) note that the Hebrew Treebank is not consistent in its treatment of punctuation, and thus evaluate their chunker only after performing normalization of chunk boundaries for punctuations. We hypothesize that, since commas and quotation marks play such an important role in the classification, performing such normalization before the training stage might be beneficial. Indeed results on the normalized corpus show improvement of about 1.0 in F score on both HEB_{Err} and HEB_{Gold}. A 10-fold cross validation experiment on punctuation normalized HEB_{Err} resulted in an F-Score of 92.2, improving the results reported by (Goldberg et al., 2006) on the same setting (91.4).

It is interesting to note that the Top4 Nouns (*e.g.*, these 4 Nouns that appear in the most Support Vectors), which are %, ש"ח/`N.I.S`, דרך/`road` and כלל/`rule`. The N.I.S and the % sign are both involved in monetary expressions which

indeed behave differently than "regular" noun phrases. The other two words compose the multi-word adverb "ב דרך כלל" ("in road of rule", "usually").

Loc=I	HEB_G	HEB_E	Loc≠I	HEB_G	HEB_E
-2	79.11	78.26	-2	93.23	91.62
-1	78.40	76.96	-1	92.75	91.86
0	92.28	90.33	0	79.64	79.44
1	78.74	76.90	1	93.46	92.33
2	78.11	76.55	2	93.41	92.18

Table 3: Results of Loc Tamperings.

We now turn to analyzing the importance of context positions (Table 3). The most important lexical feature (by far) is at position 0, that is, the word currently being classified. Back context seems to have slightly more effect than front context, and all the positions positively contribute to the decision.

Note also that when the PoS tags are fully reliable, single position lexical features on their own are more useful than in the case where PoS tags may contain errors, in which single-position lexical features are not as reliable and resorting to a combination of features is much needed.

Overall, it seems that our reliance on lexical features is not very heavy. This is very encouraging, as it suggests that as long as our POS tagger and morphological disambiguator perform well, the chunker transition to new domains is expected to be a smooth one.

4 Hard Cases and Corpus Errors

We used Anchored Learning to identify cases that are hard for the SVM classifiers to learn.

The hard cases analysis achieved by anchored learning is different from the usual error analysis carried out on observed classification errors. The traditional methods give us intuitions about where the classifier **fails to generalize**, while the method we present here gives us intuition about what the classifier **considers hard to learn**, based on the training examples alone.

4.1 Experimental Setting

Two experiments were conducted.

In the first experiment, a linear SVM model (M_{full}) was trained on the training subset of the anchored, punctuation-normalized, HEB_{Gold} corpus, with the same features as in the previous experiments, and a C value of 9,999. Corpus locations corresponding to anchors with weights >1 were inspected. There were about 120 such locations out of 4,500 sentences used in the training set. We analyzed these locations into 3 categories: corpus errors, cases that challenge the SimpleNP definition, and cases where the chunking decision is genuinely difficult to make in the absence of global syntactic context or world knowledge.

In the second experiment we aimed to understand the role of the contextual lexical features (w_i, $i \neq 0$). This is done by training 2 additional anchored linear SVM models, $M_{no-cont}$ and M_{near}. These are the same as M_{full} except for the lexical features used during training. $M_{no-cont}$ uses only w_0, while M_{near} uses w_0,w_{-1},w_{+1}.

Anchors are again used to locate the hard examples for each classifier, and the differences are examined. The examples that are hard for M_{near} but not for M_{full} are those solved by w_{-2},w_{+2}. Similarily, the examples that are hard for $M_{no-cont}$ but not for M_{near} are those solved by w_{-1},w_{+1}. Table 4 indicates the number of hard cases identified by the anchor method for each model. One way to interpret these figures, is that the introduction of features $w_{-1,+1}$ solves 5 times more hard cases than $w_{-2,+2}$.

4.2 Corpus Errors

We identified 29 hard cases related to conjunction and apposition (is the comma, colon or slash inside an NP or separating two distinct NPs). 14 of these hard cases were indeed mistakes in the corpus. This was anticipated, as heuristics were used for distinguishing appositions and conjunctive commas, and the Treebank treatment of conjunctions is also known to be somewhat inconsistent.

In order to build the Chunk NP corpus, the syntactic trees were processed to derive chunks according to the SimpleNP definition. The hard cases analysis identified 18 instances where this transformation results in erroneous chunks. For example, elided elements result in improper chunks, such as chunks containing only adverbs or only adjectives, for example:

[שבעים שנה] ו [אולי אף יותר]

An interesting manifestation of the later problem

is the case of colors. These are considered adjectives, and occur frequently with elided nouns:

[גוונים חמים] כמו [אדומים], [כתומים] ו [חומים]

It is unclear if such cases should consist of chunks, or not.

Annotation of pronouns with respect to chunk boundaries is inconsistent.

We also found 3 invalid sentences, 6 inconsistencies in the tagging of interrogatives with respect to chunk boundaries, as well as 34 other specific mistakes. Overall, more than half of the locations identified by the anchors were corpus errors. Looking for cases similar to the errors identified by anchors, we found 99 more locations, 77 of which were errors.

4.3 Challenging the SimpleNP definition

The hard cases analysis identified examples that challenge the SimpleNP definition proposed in Goldberg *et al.* (2006). The most notable cases are:

The 'et' marker : '*et*' is a syntactic marker of definite direct objects in Hebrew. It was regarded as a part of SimpleNPs in their definition. In some cases, this forces the resulting SimpleNP to be too inclusive:

[את הממשלה, הכנסת בית המשפט והתקשורת]

['et' (the government, the parliament and the media)]

Because in the Treebank the conjunction depends on '*et*' as a single constituent, it is fully embedded in the chunk. Such a conjunction should not be considered simple.

Complex determiners and quantifiers: In many cases, complex determiners in Hebrew are multiword expressions that include nouns. The inclusion of such determiners inside the SimpleNPs is not consistent. The original definition does not directly deal with the issue of complex determiners, and delegates the decision to the TreeBank annotators, which are inconsistant.

The של *preposition ('of')* marks generalized possession and was considered unambiguous and included in SimpleNPs. We found cases where 'של' causes PP attachment ambiguity:

[נשיא בית הדין] ל [משמעת] של [המשטרה]

[president-cons house-cons the-law] for [discipline] of [the police] / The Police Disciplinary Court President

Because 2 prepositions are involved in this NP, 'של' (*of*) and 'ל' (*for*), the 'של' part cannot be attached unambiguously to its head ('*court*'). This example unravels a serious problem with the definition of Hebrew NP chunks, which is further discussed in section 5.

4.4 What's Hard to Learn

The first Anchored learning experiment revealed the following phenomena that are hard for the SVM based chunker to learn:

Prepositions: the preposition של/`of` (and to a lesser extent מ/`from`), is responsible for many of the hard cases. This supports our previous intuition that של behaves differently than the rest of the prepositions. The מ related ambiguity is due to the fact that it has two uses: preposition and partitive. These are hard to distinguish, and thus both are tagged PREPOSITION by the PoS tagger.

Conjunctions, as well as commas, are generally hard to decide, this is partly due to corpus inconsistency, but also to lack of features in the local context, and the need for semantic world knowledge. For an example, consider the following sentence fragment:

מערכת העבודה השכר והאיגוד המקצועי

[The labor, salary and union system] [The labor system], [the salary] and [the union]

Some Adverbials and to a lesser extent **Adjectives** are impossible to distinguish in local context:

[ה אבדה] ל [ה משפחה] גדולה

[The loss] to [the family] (is) big / [the big family]

מוגזם לתלות את ...ב[שוק העבודה] בלבד

It would be exaggerated to attribute ... to [the labor market] alone / [the market of only labor]

Multi word idioms are common in Hebrew. For example, the adverb *unanimously* is expressed in Hebrew by the two word expression פה אחד ('one mouth'). These are hard to PoS tag (this example would be tagged as one/NUMBER mouth/NOUN), and this difficulty remains when chunking. Many Hebrew prepositions are also multi-word expressions, which include nouns and determiners. We found 10 such expressions through anchor analysis, including:

ב בת אחת	ב כל מקרה
in one time	in all cases
at once	anyway
ל כל ה יותר	כך או כך
to all the more	this-way or this-way
at most	anyway

Numeric results of second anchoring experiment are presented in Table 4.

Model	Number of hard cases ($t = 1$)	Hard cases for classifier B-I
M_{full}	120	2
M_{near}	320 (+ 200)	12
$M_{no-cont}$	1360 (+ 1040)	164

Table 4: Number of hard cases per model type.

One interpretation of this data is that the near lexical-features $w_{\pm 1}$ solve 5 times as many hard cases as the far lexical-features $w_{\pm 2}$.

Qualitative analysis of the hard cases solved by the contextual lexical features shows that they contribute mostly to the identification of chunk boundaries in cases of conjunction, apposition, attachment of adverbs and adjectives, and some multi-word expressions.

The number of hard cases specific to the B-I classifier indicates how the features contribute to the decision of splitting or continuing back-to-back NPs. Back-to-back NPs amount to 6% of the NPs in HEB_{Gold}, and 8% of the NPs in the NP chunking corpus used by (Marcus and Ramshaw, 1995). However, while in English most of these cases are easily resolved, Hebrew phenomena such as null-equatives and free word order make them harder. To quantify the difference: 79% of the first words of the second NP in English belong to one of the closed classes POS, DT, WDT, PRP, WP – categories that mostly cannot appear in the middle of base NPs. In contrast, in Hebrew, 59% are Nouns, Numbers or Proper Names. Moreover, in English the ratio of unique first words to number of adjacent NPs is 0.068, while in Hebrew it is 0.47. That is, in Hebrew, almost every second such NP starts with a different word.

These figures explain why surrounding lexical information is needed by the learner in order to classify such cases. They also suggest that this learning is mostly superficial, that is, the learner just memorizes some examples, but these will not generalize well on test data. Indeed, the most common class of errors reported in Goldberg *et al.* , 2006 are of the split/merge type. These are followed by conjunction related errors, which suffer from the same problem. Morphological features of *smixut* and agreement can help to some extent, but this is still a limited solution. It seems that deciding the [NP][NP] case is beyond the capabilities of chunking with local context features alone, and more global features should be sought.

4.5 SVM Anchored Learning vs Boosting

The intuition that "hard to learn" examples are suspect corpus errors is not new, and appears also in Abney *et al.* (1999) , who consider the "heaviest" samples in the final distribution of the AdaBoost algorithm to be the hardest to classify and thus likely corpus errors. While AdaBoost models are easy to interpret, this is not the case with SVM. Anchored learning allows us to extract the hard to learn cases from an SVM model. Interestingly, while both AdaBoost and SVM are 'large margin' based classifiers, there is less than 50% overlap in the hard cases for the two methods (in terms of mistakes on the test data, there were 234 mistakes shared by AdaBoost and SVM, 69 errors unique to SVM and 126 errors unique to AdaBoost)[5].

5 Hebrew SimpleNPs revisited

Two main motivations for text chunking are:

- Avoid attachment resolution, defering it to later stages.
- Enable easier full parsing by identifying chunks based on local factors only.

Given these two objectives, the task of defining Hebrew NP chunks is inherently problematic, mostly due to $smixut$. Let's consider the following example:

ראש המחלקה לאנרגיה של איראן

Two possible chunkings are:

1. [ראש המחלקה] ל [אנרגיה] של [איראן]
2. [ראש המחלקה ל אנרגיה] של [איראן]

[5]These numbers are for pairwise Linear SVM and AdaBoost classifiers trained on the same features.

The first case follows the current SimpleNP definition given in (Goldberg et al., 2006). In the second case the preposition ל when used as a specifier is also included in the SimpleNP.

In both cases, the correct attachment cannot be made after the chunking stage, because the second chunk does not modify the head of the first chunk, but some element of the complement.

The two possible solutions are either considering the whole expression as a single chunk (3), or following the original (non-recursive) definition of Abney (1991), Ramshaw and Marcus (1995) etc, and separating the first construct from the rest of the expression (4).

3. [ראש המחלקה לאנרגיה של איראן]
4. [ראש] [המחלקה] ל [אנרגיה] של [איראן] / [ראש] [המחלקה לאנרגיה] של [איראן]

(3) is problematic because this (a) requires attachment disambiguation while chunking, and (b) can lead to really long and complex chunks (5):

5. [ראש המחלקה לאנרגיה של איראן שדנה באנרגיה]

Notice the relative clause and VP are forced inside the NP chunk, and that this extension naturally can be made even worse (6):

6. [ראש המחלקה לאנרגיה של איראן שמתכנסת מדי שבוע ודנה במקורות אנרגיה חלופיים ועניני בטחון]

(4) is problematic, because if we treat the constructs as chunks, the result can be very fragmented chunks (7-9):

7. [ראש] [הממשלה]
8. [ראש] [ממשלת] [ישראל]
9. [משרד] [ראש] [הממשלה]

Moreover, the Hebrew NP grammar places adjectives modifying the head of a smixut at the end of the smixut expression. Chosing the non-recursive definition will result in such adjectives being left out of the NP chunk (10):

10. [ראש] [הממשלה] הפופולרי

It is clear that the non-recursive case is not a valid option, and that NP chunks should be more inclusive. On the other hand, as we've seen in (5-6), $smixut$ can "trap" arbitrarily long constructions, without proposing any good "cutting point". Such arbitrarily long constructions are not acceptable as SimpleNPs – they are not simple, and require many non-local decisions and attachment disambiguation. Indeed, finding such constructions amount to full NP parsing.

We have no choice but to decide on a "cutting criteria" that will break complex NP constructions headed by $smixut$, even if the resulting chunks cannot be combined as-is to form a complete parse tree (instead, some SimpleNPs will have an "adjunction site", and a subsequent $attacher$ process will have to take that into account). Of course, we would like to define criteria that will minimize the number of such cuttings.

Our problem is then "where to draw the line" – a balance must be met between making our SimpleNPs simple enough on the one hand, and meaningful enough on the other hand. Our current intuition is that choice (3) is a good example of such balance.

As including של in the SimpleNP is already a part of the SimpleNP definition, the only change to the original definition is allowing some forms of the ל preposition, specifically ל in a specifier usage, to be part of a SimpleNP.

Allowing such usages of ל to remain inside SimpleNPs result in SimpleNPs such as:

11. פרס נובל ל שלום
12. חתן פרס נובל ל שלום
13. נאום חתן פרס הנובל ל שלום
14. שרות טוב ל דמוקרטיה הישראלית
15. הודעות ל קרוביהם ו ל מכריהם
16. ברכות ל חג המולד
17. הדרך ל מחנה הצועניים

Of these, only the first 3 are really undebatable, and a better characterization of the cases in which to allow ל inside SimpleNPs might be desirable.

It is also not clear whether it is desirable to allow the specifier ל in all cases, or only when trapped inside a $smixut$ – *e.g.*[יום הזכרון] ל [חללי צהל] vs. [טקסי יום הזכרון ל חללי צהל].

Another cases of ל which are candidates for inclusion inside SimpleNPs are those that are attached to Adjectives, such as (18):

18. נושא לגיטימי ל וויכוח

We hope further empirical analysis will help us refine the definition of SimpleNPs in a manner that will satisfy the contradictory objectives they must fulfill: easy to classify, useful for parsing and

linguistically (both syntactically and semantically) meaningful.

References

S. Abney, R. Schapire, and Y. Singer. 1999. Boosting applied to tagging and PP attachment. *EMNLP-1999*.

Steven Abney. 1991. Parsing by Chunks. *Principle-Based Parsing: Computation and Psycholinguistics*, pages 257–278.

Steven Abney. 1996. Partial parsing via finite-state cascades. *Nat. Lang. Eng.*, 2(4):337–344.

Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 665–672, Morristown, NJ, USA. Association for Computational Linguistics.

C. Cardie and D. Pierce. 1998. Error-driven pruning of treebank grammars for base noun phrase identification. In *ACL-1998*.

Stefano Federici, Simonetta Montemagni, and Vito Pirrelli. 1998. Chunking italian. linguistic and task-oriented evaluation. In *Proc. of first international conference on language resources and evaluation*.

Yoav Goldberg and Michael Elhadad. 2007. Svm model tampering and anchored learning: A case study in hebrew np chunking. In *Proceedings of the 22st International Conference on Computational Linguistics*.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2006. Noun phrase chunking in hebrew: Influence of lexical and morphological features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 689–696, Sydney, Australia, July. Association for Computational Linguistics.

T. Kudo and Y. Matsumoto. 2000. Use of support vector learning for chunk identification. In *CoNLL-2000*.

M. Marcus and L. Ramshaw. 1995. Text Chunking Using Transformation-Based Learning. In *Proc. of the 3rd ACL Workshop on Very Large Corpora*.

Knutsson Ola, Johnny Bigert, and Viggo Kann. 2003. A robust shallow parser for swedish. In *Proc. of Nodalida'03, 14th Nordic Conference on Computational Linguistics*.

Erik F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the conll-2000 shared task: chunking. In *CoNLL-2000*.

Michael Schiehlen. 2002. Experiments in german noun chunking. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

K. Sima'an, A. Itai, Y. Winter, A. Altman, and N. Nativ. 2001. Building a tree-bank of modern hebrew text. *Traitement Automatique des Langues*, 42(2).

V. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc.

Yuqi Zhang and Qiang Zhou. 2002. Chinese base-phrases chunking. In *Proceeding of the first SIGHAN workshop on Chinese language processing*, pages 1–5, Morristown, NJ, USA. Association for Computational Linguistics.