Journal of Machine Learning Research 1 (2002) 695-719 Submitted 9/01; Published 3/02

Shallow Parsing using Noisy and Non-Stationary Training Material

Miles Osborne OSBORNE@COGSCI.ED.AC.UK
Division of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, Scotland.

Editors: James Hammerton, Miles Osborne, Susan Armstrong and Walter Daelemans

Abstract

Shallow parsers are usually assumed to be trained on *noise-free* material, drawn from the same distribution as the testing material. However, when either the training set is *noisy* or else drawn from a *different* distributions, performance may be degraded. Using the parsed Wall Street Journal, we investigate the performance of four shallow parsers (maximum entropy, memory-based learning, N-grams and ensemble learning) trained using various types of artificially noisy material. Our first set of results show that shallow parsers are surprisingly robust to synthetic noise, with performance gradually decreasing as the rate of noise increases. Further results show that no single shallow parser performs best in all noise situations. Final results show that simple, parser-specific extensions can improve noise-tolerance. Our second set of results addresses the question of whether naturally occurring disfluencies undermines performance more than does a change in distribution. Results using the parsed Switchboard corpus suggest that, although naturally occurring disfluencies might harm performance, differences in distribution between the training set and the testing set are more significant.

1. Introduction

Shallow parsers are usually automatically constructed using a training set of annotated examples (Ramshaw and Marcus, 1995, Skut and Brants, 1998, Daelemans et al., 1999a). Without loss of generality, let X be a set of word sequences and Y be a set of syntactic labels. The training set will then be a sequence of pairs of the form $\langle x_1, y_1\rangle, \ldots, \langle x_n, y_n\rangle$, where $x_i \in X, y_i \in Y$. Such a sequence would be generated by some process Q. On the basis of such a training set, a shallow parser would make predictions about future, unlabelled examples.

Arguably, significant progress in shallow parsing performance will only come from having access to massive quantities of annotated training material (Banko and Brill, 2001). Unsupervised approaches (which do not require annotated material, and so can make use of unlimited quantities of raw text), by contrast, are unlikely to yield comparable levels of performance. Now, producing annotated material on an industrial scale is highly likely to result in the introduction of significant *noise* levels and/or *distributional differences* into the training set. Therefore, successfully dealing with lower-quality (but voluminous) annotated training material will hinge upon the ability of shallow parsers to tolerate high noise levels.

Until now, the need to deal with noise has not been identified as a potential problem within the shallow parsing literature. Furthermore, issues involved with distributional differences in the training set have not been dealt with either. In this paper, we consider both of these issues -noise in the training set and distribution differences- when training a variety of shallow parsers.

Our first set of results shows that a variety of shallow parsers are surprisingly robust to synthetic noise, and only show a marked decrease in performance when the noise rates are high. Further results show that no single shallow parser performs best in all noise situations. Final results show that simple, parser-specific extensions can improve noise-tolerance.

Our second set of results addresses the question of whether naturally occurring disfluencies undermines performance more than does a change in distribution. Our results suggest that, although naturally occurring disfluencies might harm performance, differences in distribution between the training set and the testing set are more significant. Note that this conclusion is only true for noise levels that are natural, and not, for example when noise levels are higher (as might be produced when quickly trying to create large volumes of annotated material).

The rest of this paper is as follows. Section 2 briefly surveys shallow parsers from the perspective of noise-tolerance. Afterwards (Section 3) we introduce the shallow parsers used in our experiments. Next (Section 4) we present a set of simple, artificial noise models. This then leads onto a set of experiments (Section 5) where we show what happens when various parsers are trained upon noisy material, and how our various parsers can be made more noise tolerant. Section 6 deals with naturally occurring disfluencies and distributional differences. Finally, Section 7 summarises the paper.

We now look more closely at noise tolerance in a variety of shallow parsers.

2. Noise and shallow parsing

Shallow parsers (as reported in the literature) usually do not have explicit mechanisms for dealing with noise. This is probably because they are assumed to be trained upon relatively noise-free material. However, this does not mean to say that shallow parsers cannot deal with noise. For example, we see that shallow parsing systems can be made more noise tolerant either by *smoothing*, selection of a model which is 'simple', with post-processing of parsed output, or else with some combination of all three methods. Such approaches potentially might also tackle the problem of non-stationary training material.

Smoothing refers to a general set of techniques whereby a model estimated from a training set is modified in some way, such that performance upon material unlike the training material improves. Such modification is motivated on *a priori* grounds (meaning information outside of the training set is brought to bear). Since noise typically manifests as a model having a complex surface, making it smoother may remove some of the effects of noise. Note that because smoothing is motivated on an *a priori* basis, smoothing may in some circumstances actually reduce performance. We shall see two different examples of dealing with noise through smoothing in Sections 3.2 and 3.3.

Selecting a model which is 'simple' can also be seen as a technique for dealing with noise. For example, maximum entropy will select a model (from the set of possible maximum likelihood models) that has parameters which are maximally compressible. This is therefore a preference for simplicity, which in turn is similar to explicit smoothing. However, unlike smoothed maximum likelihood models (which are really maximum a posteriori models), maximum entropy is still maximum likelihood estimation, and so (in principle) nothing need be lost about the training material. Note that a preference for simplicity is nothing more than a noise-avoidance strategy, which may or may not work, depending upon the nature of the sample (Schaffer, 1993). Section 3.1 describes a shallow parser based upon maximum entropy.

Post-processing the test material, in the context of shallow parsing, was advocated by Cardie and Pierce (1998). They manually created a set of rules to correct the output of their parser. Whilst they were able to show an improvement in performance, in general, such a technique is arguably ad-hoc and cannot guarantee that all error patterns are treated. In subsequent work, Cardie et al. (1999) automated the construction of these postprocessing rules.

Given the (open) question of whether some single noise-tolerance strategy works in all domains, short of finding such an approach, an alternative method is to 'average' over whatever noise-avoidance methods one has available. The general idea is that no single approach will always work, but that when taken together, an *ensemble* of approaches can work better than any single method within that ensemble (Dietterich, 1999).

Ensemble approaches have been shown to be successful in a number of domains. For example, the current best parse selection results using the Wall Street Journal were obtained using ensemble techniques (Henderson and Brill, 2000); POS tagging is improved by ensemble methods (Brill and Wu, 1998), and for the shallow parsing domain that we consider, ensemble methods enjoy a clear advantage over single learners (Kudoh and Matsumoto, 2000).

The two key ideas behind ensemble approaches is that the individual models should be better than chance (they are 'weak' learners) and that errors made by the learners are uncorrelated. Under such circumstances, the ensemble will provably be better than any of the component models. Aslam and Decatur (1993) have proved that when the appropriate assumptions are met, ensemble learning can reduce errors, even when the training material is noisy. Ensemble learning does not always guarantee a performance improvement. Should errors be correlated (or the component learners be worse than chance) then the performance of the ensemble can be worse than the performance of the components. An example of an ensemble shallow parser is given in Section 3.4.

We now describe what we mean by shallow parsing, and introduce the shallow parsers used in our experiments.

3. Four shallow parsers

In this paper, we deal with the recovery of base phrases from the (parsed) Wall Street Journal (henceforth called WSJ). Base phrasal recovery is a well-known task, and various results exist for it. See Tjong Kim Sang and Buchholz (2000) for an overview.

In brief, the task is to annotate part-of-speech (POS) tagged sentences with non-overlapping phrases. Words in sentences are marked with a 'chunk' label, which can be either an O label (meaning the word is not in a phrase), a B label (meaning the associated word is the first word of a given phrase) or an I label (meaning the word is within some given phrase). Both B and I labels are further divided into Verbal, Nominal, Adjectival, Adverbial, Prepositional and a small variety of other minor phrasal categories. In total, there are 22 possible chunk labels. Figure 1 shows an example labelled sentence. Note that

He	reckons	the	current	account	deficit	will	narrow	to
B-NP	B-VP	B-NP	I-NP	I-NP	I-NP	B-VP	I-VP	I-VP

only	#	$	1.8	billion	in	September
I-VP	B-PP	B-NP	I-NP	I-NP	B-PP	B-NP

Figure 1: An example annotated Wall Street Journal sentence

POS labels have been suppressed.

We now present the shallow parsers used in our experiments. The motivation for using these parsers is that they are broadly representative of shallow parsers, and furthermore, help deal with a potential criticism that any results obtained from a single parser might reflect idiosyncrasies of that approach, and not more general findings.

3.1 The maximum entropy-based shallow parser

We used Ratnaparkhi's maximum entropy-based POS tagger as the basis of our maximum entropy-based shallow parser (Ratnaparkhi, 1996). In brief, the tagger uses the following exponential model:

$$P(h,t) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h,t)}$$

π is a normalising constant, $\mu, \alpha_1 \dots \alpha_k$ are parameters and $f_1 \dots f_k$ are 'features'. Each parameter α_j corresponds to a feature f_j. Each feature takes an integer values representing how many times that feature was active on an example. h_i is the *history* available when predicting POS tag t_i.

Features capture aspects of the tagging task that are deemed important when modelling. Here, features are in terms of the current word to be tagged, the previous two words, the next two words and the previous two tags. The tagger also uses features such as prefixes of words, word suffixes, etc. The interested reader should consult the original paper for a description of the features used in the tagger (Ratnaparkhi, 1996).

The parameters $\mu, \alpha_1 \dots \alpha_k$ are chosen to maximise the probability of the training set:

$$P(D \mid M) = \prod_{i=1}^{n} P(h_i, t_i)$$

D is the training set (tags and their respective histories), and M is the model (parameters and their associated weights). Parameters are estimated using *Generalised Iterative Scaling* (GIS).

The tagger is trained by taking a set of sentences, and for each word in each sentence, labelling that word with an intended POS tag. Afterwards, the tagger uses GIS to find the best set of weights to maximise the probability of seeing this data. GIS is an iterative training method, and after each iteration, the likelihood probability (equation 1) increases up to a maximum.

When tagging, the model tries to recover the most likely (unobserved) tag sequence, given a sequence of observed words.

When shallow parsing using this approach, consider the following fragment of the training set:

Word	w_1	w_2	w_3	w_4
POS Tag	t_1	t_2	t_3	t_4
Chunk	c_1	c_2	c_3	c_4

Words are w_1, w_2, w_3 and w_4, tags are t_1, t_2, t_3 and t_4 and chunk labels are c_1, c_2, c_3 and c_4. The task is to predict the chunk label for word w_2 (the current word).

In order to convince the part-of-speech tagger to shallow parse, we concatenate together the POS label of the current word, the POS label of the next word (t_3), that of the following word (t_4), and finally the last four letters of the current word (w_2). This concatenation is then used in place of a more conventional word (as expected by the tagger). In place of the tagger's POS labels, we use chunk labels. The training set is then a sequence of concatentation-chunk label pairs. The motivation for encoding shallow parsing information in this way comes from a tradeoff between context and accuracy. The more context we encode, the greater our accuracy; the more context, the sparser our features will be. Osborne (2000) gives a more detailed description of some of the issues involved with shallow parsing using this tagger. Our experiments were all run for 100 iterations of GIS (unless specified otherwise).

3.2 The memory-based shallow parser

Our memory-based shallow parser was constructed using the publically-available *TiMBL* system (Daelemans et al., 2000). Memory-based learning stores all training examples in memory. Examples are vectors of features and an associated label. At run time, an unseen example is classified in terms of how similar it is to seen examples. The k most similar training examples are retrieved from memory, and the label of the unseen example is then some function of these seen examples.

In more detail, let $D(X, Y)$ be the 'distance' between vector X and vector Y. There are many possible distance metrics, but in this work we used the default *overlap* metric:

$$D(X, Y) = \sum_{i=1}^{n} w_i \sigma(x_i, y_i)$$

$$\sigma(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases}$$

Here, x_i and y_i are the i^{th} components of the vectors X and Y. w_i is a weighting which can bias matching in various ways. We used the *gain ratio* weighting scheme (as it produced reasonable results). Together, the similarity metric and weighting scheme is called IB1-IG (Daelemans and van den Bosch, 1992).

This overlap metric prefers examples which have a minimal number of (weighted) mismatches. We find the k examples that minimise the overlap metric, and select the final label as being the label that is most common within the k examples.

Throughout our main experiments (Section 5.1), we used the IB1-IG similarity metric (as it produced acceptable results for our task; the choice of metric is not crucial for our experiments), and used $k = 3$. We also experimented with various values of k in our secondary experiments (Section 5.3).

When shallow parsing, we created vectors consisting of the current word to be labelled, the current POS tag, the previous four POS tags, and the POS tag of the next word. Better results can be obtained by using even more context (at the expense of slower testing). Since we were more concerned with running large numbers of experiments than with obtaining the best possible results, we were prepared to accept a reduced performance level (that is, our approach was about 1% less than that of Veenstra and van den Bosch (2000)). Note that this shallow parser uses different information from the maximum entropy parser. However, when there is no noise, both models yield very similar performance levels to each other.

3.3 The N-gram shallow parser

We use Brant's Markov-based POS tagger as the basis of our N-gram shallow parser (Brants, 2000). Brant's tagger selects the most likely tag sequence $t_1 \ldots t_n$ as follows:

$$\text{argmax}_{t_1 \ldots t_n} P(t_{n+1} \mid t_n) \prod_{i=1}^{n} P(t_i \mid t_{i-1}, t_{i-2}) P(w_i \mid t_i)$$

Here, w_i is a word and t_{n+1} is a tag denoting the end of a sentence.

The components of the previous equation are estimated using N-grams. The model uses linear interpolation as a smoothing device (smoothing over the term $P(t_i \mid t_{i-1}, t_{i-2})$). There is a mechanism for dealing with unknown words, based upon suffixes. The model is trained using maximum likelihood estimation, whilst the weights for linear interpolation are found using deleted interpolation.

We used the default settings of the system. In particular, this means: using statistics from singleton words when dealing with unknown words, using linear interpolation to smooth and using trigrams. Note that at times we used unigrams and bigrams instead of trigrams. These cases are clearly marked in the paper.

When shallow parsing, we used the same training set (words as concatenations of POS tags, etc) as was used when training the maximum entropy shallow parser (mentioned in Section 3.1).

Later in this paper, whenever we refer to the 'N-gram parser', as should be obvious, we mean Brant's tagger retrained as a shallow parser.

3.4 The ensemble parser

Our ensemble shallow parser used all three previously mentioned parsers, and arbitrated between them using a simple majority vote. That is, we use all parsers to label material.

For each word, we count the number of times a label was predicted by the various parsers. If a label is predicted by a majority, then we use that as the label predicted by the ensemble parser. Otherwise, we default to the decision made by one of the component parsers. As alluded to in Section 5, varying the default decision yields (marginally) different performance. Note we have not used weighted majority voting as at least one of our shallow parsers is non-probabilistic.

4. Noise models

In this Section, we introduce the four noise models that were used in our experiments. Although our models are not claimed to be realistic accounts of disfluencies as found in naturally occurring language (nor of errors made by annotators), they do allow us to introduce various types of noise into the training set, such that the parsers are stressed in different ways. Our noise models furthermore are parameterised, so we can also vary the amount of noise. Note that training on disfluent corpora (such as *Switchboard*, see Section 6) would allow us to observe the effects of realistic, naturally occurring noise, but only at a fixed rate.

Our first model (`white`) simulates classification errors (called the *classification noise* model) (Angluin and Laird, 1988). Within the `white` noise model, when training, each example received by the learner is mislabelled randomly and independently with some fixed probability. For example, the fragment shown in Figure 2 might become corrupted as shown in Figure 3.

He	reckons	the	current	account
B-NP	B-VP	B-NP	I-NP	I-NP

Figure 2: Example sentence fragment, without noise

He	reckons	the	current	account
I-NP	B-VP	O	I-NP	I-NP

Figure 3: An example annotated sentence with white noise in the labels (as produced by the `white` noise model)

Mislabelling is within the set of possible labels, and no account is taken whether the corrupted example is a well-formed example (a phrase is introduced by an appropriate beginning phrasal marker). This means that examples corrupted in this manner pose a harder learning problem than do examples which are mislabelled, yet still well-formed.

This model has been studied theoretically, and various noise-tolerant learners are known to exist for it. From a linguistic perspective, this model is artificial, and it is unlikely that any real training set would be corrupted in this manner. However, it is a useful baseline.

The second model (`filled`) introduces 'filled-pauses' into sentences (Shriberg, 1994). A filled-pause is an interjection, with no semantic content, which can occur anywhere in a sentence. For example, Figure 4 shows the insertion of a filled pause into the fragment shown in Figure 2.

He	reckons	uh	the	current	account
B-NP	B-VP	B-VP	B-NP	I-NP	I-NP

Figure 4: A sentence containing a filled pause (as produced by the `filled` noise model)

Filled pauses are introduced at a fixed and independent rate. They are assumed to simply have the same chunk label as the previous word (this adds extra noise to the sequence of labels and to the sequence of words). As such, a training set with filled-pauses contains well-formed phrases.

The third model (`repeat`) introduces simple 'repetitions' into sentences (Shriberg, 1994). Figure 5 shows an example repetition. Like filled-pauses, repetitions are introduced randomly, at a fixed rate. Repetitions are assumed to be in the same phrase as that of the previous word (this prevents the introduction of illegal sequences) and are assumed to only occur once (as multiple occurrences of the same symbol can easily be eliminated by post-processing). They are also constrained to only occur before function words.[1] Since repetitions depend upon the surface form of a sentence, they will be distributed differently from the changes made by our other models.

He	reckons	reckons	the	current	account
B-NP	B-VP	I-VP	B-NP	I-NP	I-NP

Figure 5: A sentence containing a repetition (as produced by the `repeat` noise model)

The final noise model (`all`) used all three previous models. In more detail, we applied the `repeat` model, then the `filled` model, and finally the `white` model to the training set.

As can be expected, our noise models are simple, and so arguably create unrealistic training scenarios. For example, our approach largely predicts that disfluencies are linearly distributed. Shriberg (1994) found that the rate of disfluencies in some texts were best described exponentially. The filled pause and repetition models should assign part-of-speech tags by running a part-of-speech tagger over the corrupted material.

The noise models are not that closely tied to the types of noise one might expect when manually annotating material on a massive scale. Instead, they are largely inspired by the disfluency literature. In lieu of a study of the kinds of errors made when annotating material, we conjecture that the *white* noise model might be seen as crudely approximating annotation errors. That is, the examples are noise-free, but the actual labels will be at times wrong. The other noise models may be appropriate when training on material from another genre. For example, when moving from speech to edited newswire text. Interesting future work should attempt to characterise the classes of errors made when annotating very large quantities of material, capture such errors in noise models, and then study their effects.

1. Constraining repetitions in this way is an attempt to model the fact that 40% of disfluencies in natural text behave in this manner (Shriberg, 1994).

In their defence, the noise models even if unrealistic, do allow experiments to be carried-out whereby noise-rates can be varied. Varying the noise rate cannot be done so easily using naturally occurring material.

We now present our experiments using synthetic noise in the training sets.

5. Experiments using artificial noise

Throughout these experiments, we used a training set consisting of $8,885$ sentences. This was automatically extracted from the parsed Wall Street Journal (Marcus et al., 1993). We also used a disjoint testing set ($2,011$ sentences), also drawn from the parsed Wall Street Journal. This training and testing set was the same one as used by the CoNLL00 shared task.[2]

Note that the training and testing set was re-tagged using the Brill tagger (Brill, 1992). This means that the data (with respect to the original parsed Wall Street Journal) contains tagging errors (noise). However, as both the training and testing sets were tagged in this manner, the effects of tagging errors will be factored-out. Whenever we say 'uncorrupted' material, we really mean the material as used in the CoNLL00 shared task. Future work should repeat our experiments using training and testing material as originally tagged in the Wall Street Journal.

When shallow parsing (as was previously mentioned in Section 3), the task was to assign chunk labels to tagged sentences. Evaluation was in terms of an 'f-score'. This is the (harmonic) average of recall and precision, where *recall* is the percentage of exactly matching phrases in the testing set found by the system, and *precision* is the percentage of detected phrases that are correct. Shallow parsers are usually measured using f-scores, and not in terms of tagging accuracy. For the purposes of our experiments, the actual choice of metric is not that important.

The best reported system (an ensemble) achieved an f-score of 93.84 (Kudoh and Matsumoto, 2000). In general, shallow parsing is harder than POS tagging since the correct assignment of chunk labels can depend upon much larger contexts than those that are used by POS taggers. For example, shallow parsing typically needs a context of five words and/or POS labels. POS taggers usually operate in terms of three words and/or POS tags.

As a baseline, a system which made decisions according to the most likely chunk label given a POS tag (using an uncorrupted training set) achieved an f-score of 77.07. For all noise models, and varying the rate of noise, this baseline mainly held (as shown in Figure 6). These results are the averages of 10 runs. That is, we used the same training and testing set on each run, but because our noise models make decisions on a random basis, they produced different sets of disfluencies after each run.

All other experiments were single runs (as there were so many of them). Note that at times, performance using noisy material is marginally better (at least for our baseline model) than for performance using 'clean' material. This is probably an artifact.

The reason for our baseline system having high levels of performance, even when the noise rates are high, follows from the highly peaked nature of the underlying distributions. Figure 7 shows an example distribution from the baseline parser. As can be seen, a few

2. The site `http://lcg-www.uia.ac.be/conll2000/chunking/` describes the task and contains all the training, testing and evaluation material used here.

Noise Model	Rate	F-Score	Noise Model	Rate	F-Score
`White`	25	77.10	`White`	50	77.11
`Repeat`	25	77.16	`Repeat`	50	77.96
`Filled`	25	77.06	`Filled`	50	77.06
`All`	25	77.10	`All`	50	77.98
`White`	75	77.15	`White`	100	2.93
`Repeat`	75	78.27	`Repeat`	100	78.27
`Filled`	75	77.06	`Filled`	100	77.06
`All`	75	78.23	`All`	100	3.57

Figure 6: Baseline performance using various noise models. *Rate* shows the percentage chance of a corruption occurring

Label	$P(\text{label} \mid \text{VBN})$	Count
B-ADVP	0.0002	1
I-ADVP	0.0002	1
O	0.0015	7
I-ADJP	0.0092	44
B-PP	0.0164	78
B-ADJP	0.0193	92
B-NP	0.0393	187
I-NP	0.0951	453
B-VP	0.2297	1094
I-VP	0.5891	2806

Figure 7: Distribution of chunk labels for $P(\text{label} \mid \text{VBN})$, trained upon uncorrupted WSJ

points in this distribution accounts for the majority of the probability mass. When noise is introduced by our models, these distributions become flatter. However, because the distributions are so peaked, even with a large amount of noise, these peaks are still prominent, and so performance is maintained.

5.1 Performance using noise models

Here we show the performance of our shallow parsers in the presence of various kinds of noise. Section 5.2 analyses these results.

Figure 8 shows the results using the `white` noise model. Here, as in graphs 9, 10 and 11, we show the performance of all single-model parsers, and the performance of the best ensemble parser, as the noise rate increases. We have not shown all ensemble parsers as, by-and-large, they are all quite similar to each other. It should be clear which parser relates to which curve from the labels.

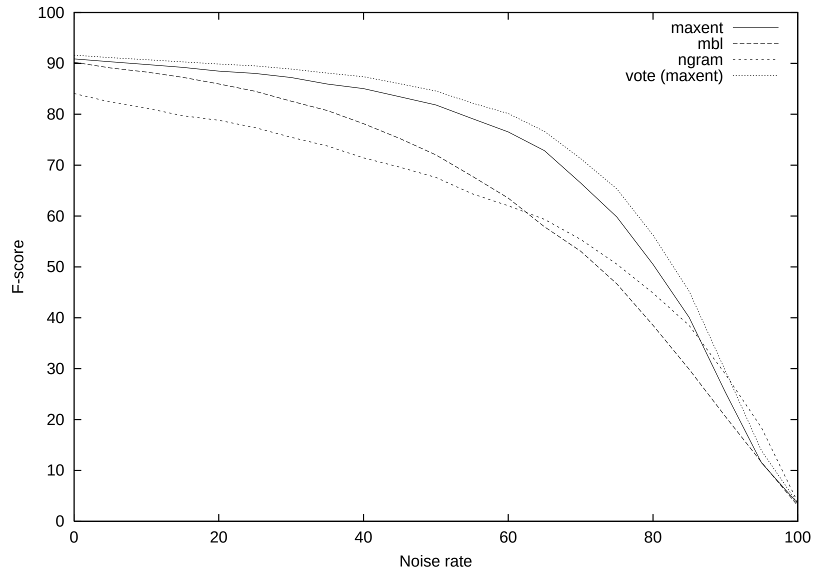

Figure 8: Results using `White` Noise Model

As can be seen, increasing the noise rate decreases the performance of all our parsers. Furthermore, we see that this degradation is relatively graceful: no parser exhibits a catastrophic performance drop as noise is introduced into the training set. When there is noise in the labels, we see that the maximum entropy parser is more robust than the memory-based parser. The N-gram parser is mid-way between the two other single-model parsers.

Turning now to the ensemble parsers (only the ensemble parser defaulting to maximum entropy is shown), firstly, we find that all three parsers usually outperform the single-model parsers. Ensemble learning can make shallow parsing more robust in the presence of white noise. Secondly, we see that defaulting to maximum entropy usually produces the best results. Finally, at 100% noise the training set cannot be distinguished from the results of flipping a coin, and so performance is minimal for all parsers.

Looking now at the results using our `filled` noise model (Figure 9), we see a different picture. The main difference from the previous results is that the memory-based parser is now the clear winner, even generally outperforming the ensemble parsers. With the various ensemble parsers, we find that again, defaulting to the behaviour of the best single parser yields marginally the best ensemble parsing results. However, we also found that the results for the other two ensemble parsers (maximum entropy default and n-gram default) produce results that are similar to each other.

The results using the `repeat` noise model (Figure 10) are different again from the previous two sets of results. Generally, we see that performance is much less affected than before.

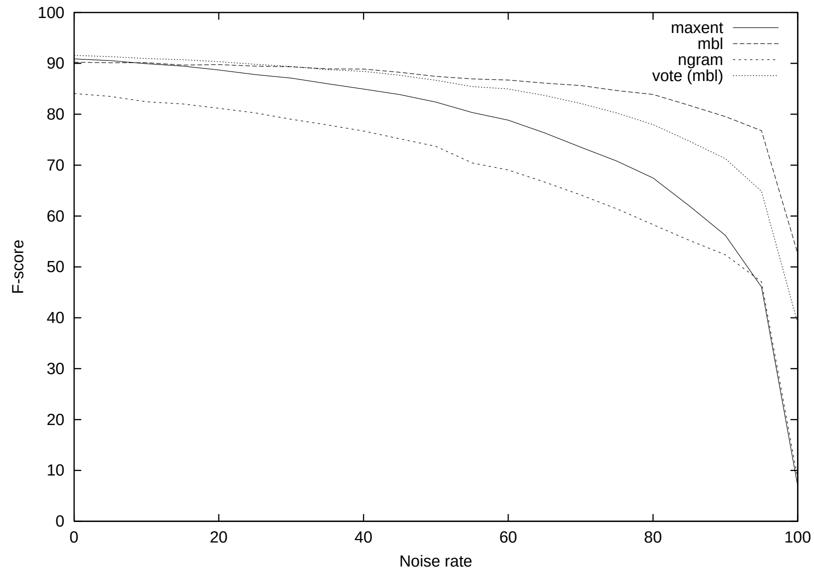

Figure 9: Results using `Filled` Noise Model

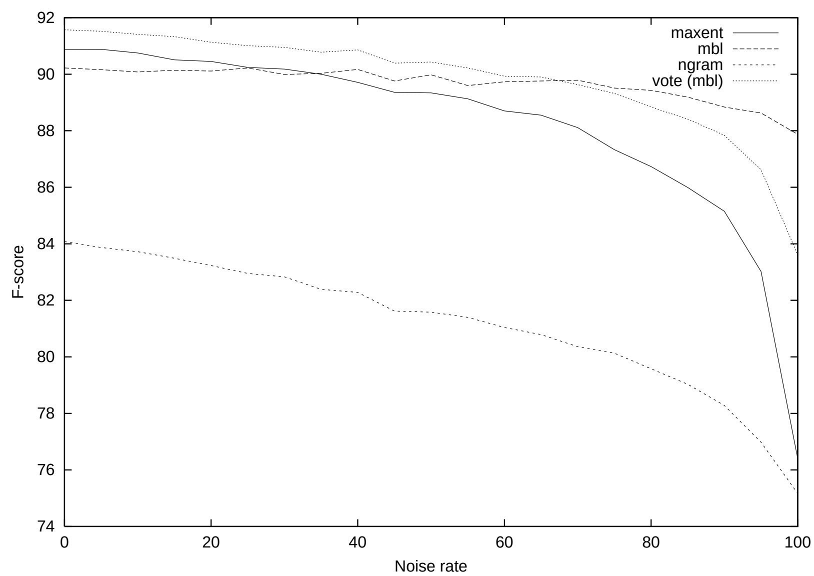

Figure 10: Results using `Repeat` Noise Model. Note the change of scale.

Our final set of results, as produced by the `all` noise model (shown in Figure 11) reveals behaviour that is again different from the other results. As shown previously in Figure 9, the memory-based parser does best out of the single model parsers. However, like the results shown in Figure 8, the ensemble parsers mainly do better than the single-model parsers. Unlike the previous results, we see that defaulting to the N-gram parser, for high noise levels, yields better results than when defaulting to the maximum entropy parser.

Figure 12 summaries which parser is best, for selected noise models.

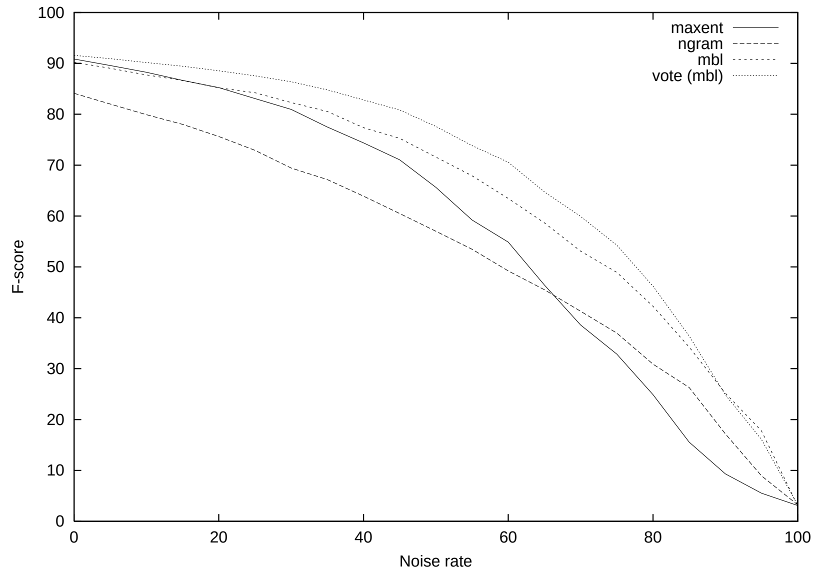

Figure 11: Results using `All` Noise Model

Rate	model	Best	model	Best	model	Best	model	Best
25	`white`	vote (MBL)	`filled`	vote (MBL)	`repeat`	vote (MBL)	`all`	vote(MBL)
50	`white`	vote (maxent)	`filled`	MBL	`repeat`	vote (MBL)	`all`	vote (MBL)
75	`white`	vote (maxent)	`filled`	MBL	`repeat`	MBL	`all`	baseline

Figure 12: Summary of results

5.2 Analysis

Here we analyse the results. The most important point to be made is that no parser is uniformly best. For example, the ensemble parser (defaulting to maximum entropy) generally yields the best results with respect to the `white` noise model. However, for the `filled` noise model, we see that the memory-based parser is usually best. The reason that we get this variation in performance is due to properties of the noise-avoidance strategies used, and, in the case of ensemble parsers, to a violation of errors not being correlated. In more detail, we see that for white noise, because the ensemble learners beat all the single model parsers, we can conclude that the errors made are uncorrelated. Hence, the individual parsers all deal with white-noise suboptimally (otherwise, there would be no gain to be made using ensemble learning). For filled pauses, we see that both maximum entropy and N-grams make correlated errors (as the ensemble parser, defaulting to memory-based learning, is worse than the single memory-based parser), and so ensemble learning does not help.

The results using noise in the labels (the `white` model) on average are worse than the results obtained using disfluencies in the examples (the `filled` and `repeat` models). This might be because the `white` model introduces more randomness into the training set than does the other models.[3] Another reason might be that supervised learners are more sensitive to noise in labels than to noise in examples.

The noise-avoidance story which appears to have emerged is that, out of a space of possible noise-reduction techniques, and for the set of parsers considered in this paper, no single method appears to work uniformly well. Theoretical support for this position is given by Wolpert and Macready (1995). Their *No Free Lunch* theorem (informally) states that no single algorithm will work uniformly well across all problems. In terms of noise, this means that an algorithm which can deal well with (for example) errors in the labels of examples may not do well with (for example) errors in the examples themselves. This also suggests that a parser which does well in the noise-free case may not do well in a noisy situation, and that some other parser may do better.

We do not conclude that any one technique is better than any other approach. For example, we see that memory-based learning is closely related to smoothing using backing-off (Zavrel and Daelemans, 1997). Hence, we could probably emulate the noise-tolerance of memory-based learning within an N-gram parser that used backing-off. This is equivalent to using some other model better suited to noise-tolerance than the one we did use.

However, returning to the *No Free Lunch Theorem*, we conjecture that no single parser will be uniformly superior in all noise scenarios. So, if we made the N-gram parser tolerant to filled-pauses, we would expect it to perform worse on some other problem. Section 5.3 illustrates this point.

In summary, we see that robustness to noisy training material is a many-faceted problem, and it is unlikely that any single strategy will work well uniformly. Instead, we advocate understanding the properties of the noise-avoidance strategy used by the parser and determining whether it is suitable for some particular domain. We do not believe in a silver-bullet algorithm which will work well for any given problem.

5.3 Parser-specific extensions

Here we consider, in the light of our previous experiments, whether we can make our parsers more robust to noise. We restrict ourselves to what can be easily achieved within the framework of a particular parser. So, we do not build new parsers from scratch; instead we vary any available parameters that might improve results. This restriction is natural in that one is more likely to try to get the best results from a given off-the-shelf parser than try to build one from scratch.

Note that we are not primarily interested in trying to maximise the performance of a system for a given noisy training set. Instead, we wish to gain an insight into how parser-specific extensions behave across a variety of settings. As such, we report system performances for a variety of noise models and for different noise levels. We therefore have not used a held-out set to optimise the various parser extensions. Clearly however, in a practical setting, one would want to do this.

Turning to the maximum entropy parser, within the confines of the implementation, the only parameter we can vary is the number of training iterations allowed to fit the

3. An anonymous reviewer made the point that because this noise model does not guarantee that examples are well-formed, the performance drop may be also due to the fact that the hypothesis space is potentially much larger. Hence, it would be possible to reduce this drop by making sure that examples are well-formed, possibly by some post-processing.

model. As was mentioned in Section 3.1, after each iteration the model fits the training material more closely. In general (modulo overfitting), after each iteration, performance also increases. Now, roughly speaking, after a few iterations, the model fits aspects of the data of which there is ample evidence. As the number of iterations increases, the model fits more specific 'patterns'. Should it be possible to differentiate noise from the underlying, uncorrupted machinery producing the data in this manner, then it is possible that early stopping (running for only a few iterations rather than for many iterations) would yield better performance than when halting after carrying out many iterations.

Figure 13 shows the results we obtained when varying the number of iterations (using the `all` noise model). The columns marked 'Iter' refer to the number of iterations used to train a model.

Rate	Iter	F-score	Iter	F-score	Iter	F-score	Iter	F-score	Iter	F-score
0	20	89.89	40	90.54	60	90.67	80	90.81	100	90.87
10	20	88	40	88.54	60	88.62	80	88.44	100	88.32
20	20	85.56	40	85.81	60	85.7	80	85.46	100	85.24
30	20	82.1	40	82.01	60	81.37	80	80.83	100	80.75
40	20	75.47	40	75.54	60	74.94	80	74.64	100	74.51
50	20	68.25	40	67.44	60	66.86	80	66.7	100	66.42
60	20	57.26	40	55.38	60	54.62	80	54.31	100	54.12
70	20	45.22	40	40.81	60	39.55	80	39.23	100	39.2
80	20	30.89	40	25.38	60	23.74	80	23.14	100	22.9
90	20	14.34	40	11.73	60	10.94	80	10.62	100	10.47
100	20	3.19	40	3.29	60	3.33	80	3.46	100	3.56

Figure 13: Results for the maximum entropy parser, using early stopping, with the `all` noise model

As can be seen, early stopping does improve results: as the noise rate increases, it is better to use fewer and fewer iterations.

Next, we considered whether we could make the memory-based parser more resistant to noise in the labels. The TiMBL implementation allows one to vary the number of nearest neighbours that are used when classifying unseen examples. When $k = 1$, the system does no smoothing over labels, and bases its decision upon just the single best matching example. As k increases, the system becomes less sensitive to individual labels. So, varying k ought to make the parser more robust to labelling errors.

Figure 14 shows the results obtained when training over material again corrupted by the `all` noise model. For each noise level, we varied k and recorded the results.

Clearly, varying the number of nearest neighbours (k) improves results as the amount of noise increases. Note that what is best when training upon material with no added noise (a rate of 0) is not best when training upon material that is noisy. Also, we see that with no noise, the best value of k is not 1. This suggests that the Wall Street Journal is noisy (a well known result). Another possible reason is that this is some strange result of the particular choice of features we used.

Rate	k	F-score	k	F-score	k	F-score	k	F-score	k	F-score	k	F-score
0	1	89.44	3	90.22	5	89.78	7	88.96	9	88.16	11	87.58
10	1	78.26	3	87.74	5	87.79	7	87.36	9	86.56	11	86.09
20	1	66.86	3	84.97	5	85.94	7	85.86	9	85.83	11	85.42
30	1	57.51	3	82.93	5	85.14	7	85.02	9	84.65	11	84.04
40	1	46.96	3	77.89	5	82.92	7	83.89	9	84.14	11	83.61
50	1	36.46	3	71.35	5	80.48	7	82.13	9	82.19	11	81.77
60	1	29.67	3	64.6	5	77.18	7	80.81	9	80.73	11	80.37
70	1	22.44	3	52.97	5	70.86	7	77.05	9	78.69	11	78.17
80	1	16.99	3	42.89	5	61.93	7	71.84	9	75.28	11	75.18
90	1	11.09	3	25.18	5	41.38	7	54.18	9	63.22	11	65.75
100	1	2.93	3	2.77	5	2.62	7	2.74	9	2.09	11	1.8

Figure 14: Results for the memory-based parser, varying k and using the `all` noise model

Our results can be seen as being largely contrary to the related work of Daelemans et al. (1999b). They also varied k for a range of language problems. In general, they found that increasing k (from 1) decreased performance. Partly on the basis of this, they concluded that one should not abstract (smooth) from the data. Clearly we see that this argument only holds when using clean training material.

Finally, we considered whether we could improve the results of the N-gram parser. The implementation we use allows one to vary the order of the Markov model.[4] That is, we can use either unigrams (order 1), bigrams (order 2) or else trigrams (order 3). Clearly, the lower the order, the simpler the model. Simple models can sometimes help deal with the effects of noise (as was argued in Section 2).

As before, we corrupted the training set with varying rates of noise (using the `all` model) and estimated models using these corrupted training sets. We also varied the order of the model. Figure 15 summarises our results.

As expected, we see that with increasing amounts of noise, it becomes better to use simpler models. Note that we could have further simplified our models by reducing the information present within a 'word'. If we recall, the N-gram parser is trained using words which are concatenations of (for example) POS tags. By reducing the information content of a word, we would also simplify the model, and so presumably improve upon our results. The baseline parser (which only considers POS tags of a unigram model) can be seen as being such a simplified approach.

In summary, we see that in theory, all of our single-model parsers can be made more noise-resistant. Furthermore, we see that tackling noise means abstracting from the training material. For example, early stopping prevents maximum entropy from modelling all regularities in the data; increasing the size of k in memory-based learning forces the system to abstract away from actual examples; reducing the order of a Markov model means making

4. Note that it is also possible to use less specific features in an N-gram model. This would also simplify the resulting model, and so make it less sensitive to noise. An interesting set of experiments would be to see how this approach compares with varying the order of the model.

Rate	Model order	F-score	Model order	F-score	Model order	F-score
0	1	75.45	2	83.86	3	84.10
10	1	74.20	2	80.21	3	80.23
20	1	71.52	2	76.22	3	75.44
30	1	69.21	2	71.52	3	69.86
40	1	65.98	2	66.37	3	64.04
50	1	60.35	2	59.48	3	56.64
60	1	54.14	2	52.54	3	49.85
70	1	46.43	2	43.71	3	41.42
80	1	36.77	2	33.77	3	31.81
90	1	20.88	2	17.79	3	17.03
100	1	2.15	2	2.42	3	2.61

Figure 15: Results for the N-gram-based parser, varying the model order and using `All` noise model

the parser ignore more and more of the context in the training set. In practice (trying to optimise a parser for noise in realistic setting), our noise-avoidance strategies may become less useful. Clearly this is an empirical matter.

In the next Section, we look at what happens when the training set contains naturally occurring disfluencies.

6. Experiments using Switchboard

Since our previous experiments all used naturally occurring language corrupted by artificial noise, we also trained our shallow parsers using realistically disfluent text. This allows us to investigate naturally occurring noise. We did not have access to (noisy) material produced by people annotating large quantities of text and so could not carry out experiments measuring noise related to annotator errors.

The WSJ is not disfluent (being edited newswire material) so studying the effects of naturally occurring disfluencies requires non-edited material. The parsed *Switchboard Corpus* (henceforth called SWBD) consists of manually parsed telephone conversations between pairs of people. Since it is composed of conversational speech, it contains numerous disfluencies (such as interruptions, incomplete sentences etc). Training upon SWBD would therefore provide complementary results to those found using synthetic noise. However, SWBD is a different genre from WSJ, and so phrasal distributions will be different between these two domains. We therefore have to differentiate between the effects of disfluencies and distributional variations before we can come to any conclusions about the effects of naturally occurring disfluencies on shallow parsing.

For the experiments here, we used all files in section 2 of the SWBD distribution. Using the same script (from the CoNLL website), we created a training set that consisted of utterances marked with POS tags and chunk labels. Figure 16 shows an example SWBD utterance annotated in this manner (POS labels have been suppressed). There were 35,706 sentences in our SWBD training set.

And	,	um	,	she	had	a	fall
O	O	B-INTJ	O	B-NP	B-VP	B-NP	I-NP

Figure 16: An example SWBD annotated utterance

Using the same parser configurations described in Section 5, and when testing using the same testing material (namely uncorrupted parsed WSJ material), we obtained the results shown in Figure 17. Unsurprisingly, these results are all worse than when training upon uncorrupted WSJ material (as shown, for example, in the top row of Figure 8).

Parser	F-score	Parser	F-score
Max	82.97	Vote (Max)	84.62
TNT	70.73	Vote (TNT)	84.33
MBL	83.20	Vote (MBL)	85.08
Baseline	65.86	-	-

Figure 17: Results when training upon SWBD

In order to determine whether this performance drop was due to disfluencies, or else due to a change in domain (or both), we ran further experiments. The next Section (6.1) examines the influence of disfluencies, whilst Section 6.2 looks at distributional differences between SWBD and WSJ.

6.1 Estimating the influence of noise

The first experiment tried to see if we could reduce noise using one of the techniques mentioned previously in Section 5.3. If we could reduce the effects of noise, then this would suggest that SWBD contained significant levels of noise, which in turn was contributing towards the performance drop. This assumes that there is a connection between the experimental setup in Figure 14 and 18. Figure 18 shows the results obtained when training the memory-based parser and varying k. As can be seen, the best performance occurs when k is 3. Comparing these results with those obtained using artificial noise (and the WSJ), Figure 14, we see evidence that the noise rate is less than 10%.[5] That is, for Figure 14, k being 3 is best when the noise rate is less than 10%. Noise may therefore not be the significant factor why there is a performance drop.

These results are not sufficient in themselves, so we also looked at what happened when annotated disfluencies were removed from SWBD. If these disfluencies were significantly contributing towards errors, then we might expect that this 'cleaning' of the data would

5. As an anonymous reviewer pointed out, caution should be exercised here. We do not know for sure the relationship between Switchboard noise and our noise models. This means that it still remains to be seen exactly how the k level relates to noise levels. One way of dealing with this issue would be to try and model naturally occurring noise more closely and then see whether, for varying levels of noise, there was a relationship with k.

k	1	3	5	7	9	11
F-score	80.35	83.20	82.63	81.68	80.94	80.32

Figure 18: Results when training upon SWBD, using the memory-based parser, varying k

improve our results. We removed any words that were marked as being an interjection from the training material. As can be seen in Figure 19, if anything, removing interjections harms performance. The reason for this lack of improvement is that after removing interjections, phrases are made contiguous that otherwise would have been separated by interjections.

Parser	F-score	Parser	F-score
Max	81.90	Vote (Max)	83.77
TNT	70.47	Vote (TNT)	83.57
MBL	82.70	Vote (MBL)	84.58
Baseline	72.24	-	-

Figure 19: Results when training upon SWBD with some disfluencies removed

Figure 20 shows a distribution of the baseline parser (when trained upon the usual SWBD training set). Comparing this distribution with the one obtained when the baseline parser was trained on WSJ material (Figure 7), we see that there are differences. For example, in Figures 7 and 20, we see that in SWBD, words tagged with the POS label VBN are more likely to be within a verbal chunk than to start a verbal chunk in WSJ (as was the case in the WSJ corpus). Furthermore, we see that the distribution for SWBD is not significantly more spread-out (more uniform) than the distributions for WSJ. If disfluencies were strongly present, and were uniformly distributed, then we might expect to see this.

Label	P(label | VBN)	Count
I-CONJP	0.0002	1
B-NP	0.0035	18
O	0.0037	19
B-PP	0.0066	34
I-NP	0.0072	37
I-ADJP	0.0093	48
B-ADJP	0.0231	119
B-VP	0.2936	1512
I-VP	0.6527	3361

Figure 20: Distribution of chunk labels for P(label | VBN), trained upon SWBD

In summary, we are lead to believe that naturally occurring disfluencies are not that harmful to performance. The next set of experiments reinforces this opinion.

6.2 Estimating the effects of distributional differences

In order to determine whether the performance drop was due to distributional differences, we considered the issue of what happened when the SWBD training material was augmented with various amounts of WSJ training material. If SWBD simply lacked phrasal patterns found in WSJ, then we would expect to find a significant improvement should we simply add that missing information.

Figure 21 shows what happens when various randomly selected amounts of WSJ training are added to the SWBD material. There were $120k$ tokens (words or punctuation marks) in the SWBD material. The total WSJ training set consisted of $211k$ tokens.

Here, the column marked 'level' shows the number of WSJ tokens added (in thousands). So, a level of 0 indicates that the training material is entirely SWBD, whilst a level of 211 shows that the training material consisted of all of SWBD and the WSJ material. As can be seen, with only a small amount of WSJ material, performance significantly increases. Also, when using all material, we see that performance is very close to the performance obtained when using just the uncorrupted WSJ material (as shown in Figure 8).

Now, these results may suggest that SWBD simply lacks phrases found in WSJ, and that even a moderate amount of material containing such information is better than nothing. However, it is also possible that performance may simply be increasing due to more accurate statistics. To deal with this issue, we also used, in addition to section 2, section 3 of SWBD. Figure 22 shows our results. The training set roughly doubled in size, and now consisted of 61, 515 SWBD sentences.

Level	Max	MBL	TNT	Vote (Max)	Vote (MBL)	Vote (TNT)
0	82.31	82.83	69.99	84.15	84.66	83.89
21	86.80	86.14	77.60	87.84	88.24	87.81
42	87.91	87.45	79.46	88.80	89.18	88.87
63	89.01	88.07	80.49	89.80	90.07	89.79
84	89.27	88.29	81.11	89.96	90.08	89.89
105	89.76	88.80	81.73	90.27	90.31	90.23
126	90.19	89.02	82.02	90.65	90.71	90.61
147	90.17	89.00	82.92	90.68	90.68	90.56
168	90.55	89.39	82.91	91.01	91.05	90.92
189	90.72	89.43	83.41	91.03	91.02	90.93
211	90.84	89.63	83.65	91.03	91.03	90.92

Figure 21: Results when training upon SWBD and varying randomly sampled quantities of *WSJ*

As can be seen (by comparing Figures 22 and 17), our results only marginally increase, and so the results obtained in Figure 21 are not due to more accurate estimation of models due to training upon more material.

An error analysis of the mistakes made when training just on SWBD material (using the best ensemble parser), and not made when training upon WSJ material, found that:

- Noun phrases in the WSJ are predicted as being too short. For example, we see:

Parser	F-score	Parser	F-score
Max	83.35	Vote (Max)	84.16
TNT	71.37	Vote (TNT)	84.98
MBL	83.39	Vote (MBL)	85.16
Baseline	63.71	-	-

Figure 22: Results when training upon double the quantity of SWBD material

Rockwell	International	Corp.	's	Tulsa	unit	said	...
	I-NP / B-NP	I-NP / B-NP					

Here, the words *International* and *Corp* are labelled as being the start of a base noun phrase, when in reality they should all be within the same base noun phrase.

- Base noun phrases with non-nominal premodifiers are seen as being phrases other than base noun phrases:

...	it	had	operating	profit	of	\$	10	million
			B-NP / I-VP	I-NP / B-NP				

- There was little evidence of 'strange' parsing decisions, as one would expect if disfluencies were responsible for parsing errors.

In conclusion then, we believe that the disfluencies in SWBD, whilst still a factor, are less harmful to performance than the change of domain (speech to edited text). Alternatively, this means that even if SWBD was edited and all disfluencies were removed, should we train upon that, we would not expect to obtain a significant performance increase. When the task is shallow parsing newswire material, it is better to annotate more newswire material than to train upon material drawn from a significantly different distribution. Gildea (2001) made a related point when training full-scale parsers using the Brown and WSJ parsed treebanks.

7. Comments

We mentioned in the introduction that shallow parsing could in principle be undermined by either noise in the training set, or else by distributional differences between the training set and the testing set. The first set of results showed that the underlying distributions are such that shallow parsing is inherently noise-tolerant, and that only large quantities of noise will significantly undermine performance. However, should one wish to improve performance when noise was present, then simple extensions of our basic parsers help. Our experiments showed that different kinds of noise favoured different parsers, and that in general, no single technique emerged as being best in all situations. When dealing with a new problem (with unknown noise), the best strategy is probably to consider a range of techniques, optimise them using a held-out set, and then decide which one to use on the basis of results. The alternative, of taking a favourite approach and then hoping that is will perform well is almost certainly likely to lead to suboptimal results.

We do not wish to comment upon the individual parsers (for example, arguing that memory-based learning is better than maximum entropy). The reason for abstaining follows from our results, which tells us that even if some parser appeared to do well in many situations, we should not conclude that it would do well in some future situation. However, we were pleased with the performance of memory-based learning, and suggest that it might be well suited for domains such as discourse parsing, which is known to be noisy.

The experiments with SWBD suggested that a mismatch in distribution between examples in the training set and the testing set undermined performance more than did disfluencies. Furthermore, we concluded that should one want to improve upon the performance of shallow parsing, it is better to annotate more examples, drawn from the target distribution (which in this case is the WSJ) than to train upon additional material drawn from other distributions. Language is non-stationary, and pretending otherwise is ill-advised.

Drawing together the results from WSJ and SWBD, and with the caveat that the noise models we used are artificial and so not necessarily indicative of realistically found noise, one might conclude that naturally occurring noise is not that much of an issue, and that instead, making sure that the training set and testing sets are both drawn from the same distribution is the key factor. With current treebanks, this is almost certainly true. However, should one want to greatly increase the amount of annotated material in some industrial manner, then undoubtedly such newly created material would become noisier. Under such circumstances, and again given the caveat regarding the relationship between our noise models and naturally occurring noise, the techniques mentioned in this paper would become relevant.

In this paper, we only considered shallow parsing using base phrases. Further work should consider what happens when parsing is deeper. We conjecture that the distributions associated with deep parsing would continue to be sharply peaked. Consequently, we would expect similar noise-performance relationships to those reported here.

Acknowledgments

We would like to thank Maria Wolters and Mark Core for discussions about disfluencies, the authors of the various systems used in this work, and the three anonymous reviewers for very helpful comments.

References

Dana Angluin and Philip Laird. Learning from Noisy Examples. *Machine Learning*, 2(4): 343–370, 1988.

Javed A. Aslam and Scott E. Decatur. General Bounds on Statistical Query Learning and PAC Learning with Noise via Hypothesis Boosting. In *Proc. of the 34rd Annual Symposium on Foundations of Computer Science*, pages 282–291. IEEE Computer Society Press, Los Alamitos, CA, 1993. URL `citeseer.nj.nec.com/aslam93general.html`.

Michele Banko and Eric Brill. Scaling to Very Very Large Corpora for Natural Language Disambiguation. In *Proceedings of ACL 2001*, Toulouse, 2001.

T. Brants. TnT – a Statistical Part-of-Speech Tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000*, Seattle,WA., May 2000.

Eric Brill. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92,* 3^{rd} *Conference on Applied Natural Language Processing*, pages 152–155, Trento, IT, 1992. URL `citeseer.nj.nec.com/article/brill92simple.html`.

Eric Brill and Jun Wu. Classifier Combination for Improved Lexical Disambiguation. In Christian Boitet and Pete Whitelock, editors, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 191–195, San Francisco, California, 1998. Morgan Kaufmann Publishers. URL `citeseer.nj.nec.com/brill98classifier.html`.

Claire Cardie, Scott Mardis, and David Pierce. Combining error-driven pruning and classification for partial parsing. In *Proceedings of the* 16^{th} *International Conference on Machine Learning*, pages 87–96. Morgan Kaufmann, San Francisco, CA, 1999. URL `citeseer.nj.nec.com/cardie99combining.html`.

Claire Cardie and David Pierce. Error-Driven Pruning of Treebank Grammars for Base Noun Phrase Identification. In *Proceedings of the* 17^{th} *International Conference on Computational Linguistics*, pages 218–224, 1998.

W. Daelemans, S. Buchholz, and J. Veenstra. Memory-based shallow parsing. In *Proceedings of CoNLL*, Bergen, Norway, 1999a.

W. Daelemans and A. van den Bosch. Generalization performance of backpropagation learning on a syllabification task. In *Proceedings of the 3rd Twente Workshop on Language Technology*, 1992. URL `citeseer.nj.nec.com/daelemans92generalization.html`.

Walter Daelemans, Antal Van Den Bosch, and Jakub Zavrel. Forgetting Exceptions is Harmful in Language Learning. *Machine Learning*, 34:11, 1999b. URL `citeseer.nj.nec.com/daelemans99forgetting.html`.

Walter Daelemans, Jakub Zavrel, Ko van der Sloat, and Antal van den Bosch. TiMBL: Tilburg Memory Based Learner, Version 3.0, Reference Guide. Technical Report ILK Technical Report 00-01, Tilburg, 2000. Available from http://ilk.kub.nl/software.html.

T. G. Dietterich. Ensemble Methods in Machine Learning. In J. Kittler and F. Roli, editors, *First International Workshop on Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 1–15, New York, 1999. Springer Verlag.

Daniel Gildea. Corpus Variation and Parser Performance. In *Proceedings of 2001 Conference on Empirical Methods in Natural Language*, Pittsburgh, PA, 2001.

John C. Henderson and Eric Brill. Bagging and Boosting a Treebank Parser. In *Proceedings of the* 1^{st} *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*, pages 34–41, 2000.

Taku Kudoh and Yuji Matsumoto. Use of Support Vector Learning for Chunk Identificatio. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, 2000. http://lcg-www.uia.ac.be/conll2000/chunking/.

M. Marcus, S. Santorini, and M. Marcinkiewicz. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL `citeseer.nj.nec.com/marcus93building.html`.

Miles Osborne. Shallow Parsing as Part-of-Speech Tagging. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, 2000. http://lcg-www.uia.ac.be/conll2000/chunking/.

Lance A. Ramshaw and Mitchell P. Marcus. Text Chunking Using Transformation-Based Learning,. In *Proceedings of the* 3^{rd} *ACL Workshop on Very Large Corpora*, pages 82–94, June 1995.

Adwait Ratnaparkhi. A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of Empirical Methods in Natural Language*, University of Pennsylvania, May 1996. Tagger: `ftp://ftp.cis.upenn.edu/pub/adwait/jmx`.

Cullen Schaffer. Overfitting Avoidance as Bias. *Machine Learning*, 10:153–178, 1993.

Elizabeth Shriberg. *Preliminaries to a Theory of Speech Disfluencies*. PhD thesis, University of California at Berkeley, 1994.

W. Skut and T. Brants. A Maximum Entropy Partial Parser for Unrestricted Text. In *Proceedings of the 6th ACL Workshop on Very Large Corpora*, 1998.

Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, 2000. http://lcg-www.uia.ac.be/conll2000/chunking/.

Jorn Veenstra and Antal van den Bosch. Single-Classifier Memory-Based Phrase Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, 2000. http://lcg-www.uia.ac.be/conll2000/chunking/.

David H. Wolpert and William G. Macready. No Free Lunch Theorems for Search. Technical Report SFI-TR-95-02-010, 1995. URL `citeseer.nj.nec.com/wolpert95no.html`.

Jakub Zavrel and Walter Daelemans. Memory-Based Learning: Using Similarity for Smoothing. In *Proceedings of the* 35^{th} *Annual Meeting of the Association of Computational Linguistics*, pages 436–443, Madrid, July 1997.