Training Dependency Parser Using Light Feedback

Avihai Mejer
Department of Electrical Engineering
Technion-Israel Institute of Technology
Haifa 32000, Israel
amejer@tx.technion.ac.il

Koby Crammer
Department of Electrical Engineering
Technion-Israel Institute of Technology
Haifa 32000, Israel
koby@ee.technion.ac.il

Abstract

We introduce lightly supervised learning for dependency parsing. In this paradigm, the algorithm is initiated with a parser, such as one that was built based on a very limited amount of fully annotated training data. Then, the algorithm iterates over unlabeled sentences and asks only for a single bit of feedback, rather than a full parse tree. Specifically, given an example the algorithm outputs two possible parse trees and receives only a single bit indicating which of the two alternatives has more correct edges. There is no direct information about the correctness of any edge. We show on dependency parsing tasks in 14 languages that with only 1% of fully labeled data, and light-feedback on the remaining 99% of the training data, our algorithm achieves, on average, only 5% lower performance than when training with fully annotated training set. We also evaluate the algorithm in different feedback settings and show its robustness to noise.

1 Introduction

Supervised learning is a dominant paradigm in machine learning in which a prediction model is built based on examples, each of which is composed of inputs and a corresponding *full* annotation. In the task of parsing, examples are composed of sentences in some language and associated with full parse trees. These parse trees are often generated by human annotators. The annotation process is complex, slow and prone to mistakes as for each sentence a full correct feedback is required.

We describe light-feedback learning which suits learning problems with complex or structured output, like parsing. After building an initial classifier, our algorithm reduces the work of the annotator from a full annotation of the input sentence to a *single* bit of information. Specifically, it provides the annotator with two alternative parses of the input sentence and asks for the single bit indicating which of the alternatives is better. In 95% of the sentences both alternatives are identical except for a single word. See Fig. 2 for an illustration. Thus, the work of the annotator boils down to deciding for some specific word in the sentence which of two possible words should be that word's head.

We show empirically, through simulation, that using only 1% of the training set with full annotation, and the remaining 99% with light annotation, our algorithm achieves an average accuracy of about 80%, only 5% less than a parser built with full annotated training data. These results are averaged over 14 languages. With additional simple relaxations, our algorithm achieves average accuracy of 82.5%, not far from the performance of an algorithm observing full annotation of the data. We also evaluate our algorithm under few noise settings, showing that it is resistant to noise, with a decrease of only 1.5% in accuracy under about 10% feedback noise. We defer a discussion of related work to the end of the paper.

2 Dependency Parsing and Parsers

Dependency parsing of a sentence is an intermediate between shallow-parsing, in which a given sentence is annotated with its part-of-speech, and between a full structure over the sentence, such as the ones defined using context-free grammar. Given a sentence with n words a parse tree is defined by constructing a single directed edge outgoing from each word to its head, that is the word it depends on according to syntactic or semantic rules. Additionally, one of the words of the sentence must be labeled as the root of the tree. The choice of edges is restricted to induce trees, i.e. graphs with no loops.

Dependency parsers, such as the MSTParser of (McDonald et al., 2005), construct directed edges between words of a given sentence to their arguments. We focus on non-projective parsing with non-typed (unlabeled) edges. MSTParser produces a parse tree for a sentence by constructing a full directed graph over the words of the sentence with weighted edges, and then outputting the maximal spanning tree (MST) of the graph. Given a true parse tree (aka as gold labeling) and a predicted parse tree $\hat{\boldsymbol{y}}$, we evaluate the latter by counting the number of words that are in agreement with the true labeling.

The MSTParser maintains a linear model for setting the weights of the edges of the full graph. Given the input sentence $\boldsymbol{x}$ the parser sets the weight of the edge between words x_i and x_j to be $s(i,j) = \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, i, j)$ using a feature function $\boldsymbol{f}$ that maps the input $\boldsymbol{x}$ and a pair of possibly connected words into R^d. Example features are the distance between the two words, words identity and words part-of-speech. The goal of the learning algorithm is to choose a proper value of $\boldsymbol{w}$ such that the induced tree for each sentence $\boldsymbol{x}$ will have high accuracy.

Online Learning: MSTParser is training a model by processing one example at a time using online learning. On each round the algorithm receives a new sentence $\boldsymbol{x}$ and the set of correct edges $\boldsymbol{y}$. It then computes the score-value of all possible directed edges, $s(i,j) = \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, i, j)$ for words i, j using the *current* parameters $\boldsymbol{w}$. The algorithm is computing the best dependency tree $\hat{\boldsymbol{y}}$ of this input $\boldsymbol{x}$ defined to be the MST of the weighted complete directed graph induced from the matrix $\{s(i,j)\}$. It then uses the discrepancy between the true parse tree $\boldsymbol{y}$ and the predicted parse tree $\hat{\boldsymbol{y}}$ to modify the weight vector.

MSTParser specifically employs the MIRA algorithm (Crammer et al., 2006) to update the weight vector $\boldsymbol{w}$ using a linear update,

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha \left(\sum_{(i,j) \in \boldsymbol{y}} \boldsymbol{f}(\boldsymbol{x}, i, j) - \sum_{(i,j) \in \hat{\boldsymbol{y}}} \boldsymbol{f}(\boldsymbol{x}, i, j) \right) \quad (1)$$

for input-dependent scalar α that is defined by the algorithm. By construction, correct edges (i,j), that appear both in the true parse tree $\boldsymbol{y}$ and the predicted parse tree $\hat{\boldsymbol{y}}$, are not affecting the update, as the terms in the two sums of Eq. (1) cancel each other.

3 Online Learning with Light Feedback

Supervised learning is a very common paradigm in machine learning, where we assume having access to the correct full parse tree of every input sentence. Many algorithms, including MSTParser, explicitly assume this kind of feedback. Supervised learning algorithms achieve good performance in dependency parsing, but they come with a price. Human annotators are required to fully parse each and every sentence in the corpora, a long, tedious and expensive process, which is also prone to mistakes. For example, the first phase of the famous penn tree bank project (Marcus et al., 1993) lasted three years, in which annotators corrected outputs of automated machines in a rate of 475 words per hour. For supervised learning to be successful, typically a large set of thousands instances is required, which translates to a long and expensive annotation phase.

Binary or multi-class prediction tasks, such as spam filtering or document classification, are simple in the sense that the label associated with each instance or input is simple. It is either a single bit indicating whether the input email is spam or not, or one of few values from a fixed predefined set if topics. Dependency parsing is more complex as a decision is required for every word of the sentence, and additionally there is a global constraint of the parse being a tree.

In binary classification or multi-class problems it is only natural to either annotate (or label) an example, or not, since the labels are atomic, they cannot be decomposed to smaller components. The situation is different in structured tasks such as dependency parsing (or sequence labeling) where each instance is constructed of many elements that each needs to be annotated. While there are relations and coupling between the elements it is possible to annotate an instance only partially, such as provide a dependency edge only to several words in a sentence.

We take this approach to the extreme, and consider (for now) that for each sentence only a single bit of labeling will be provided. The choice of what bit to require is algorithm and example dependent. We propose using a *light feedback* scheme in order to significantly reduce annotation effort for dependency parsing. First, a base or initial model will be learned from a *very small* set of fully annotated examples, i.e. sentences with full dependency information known. Then, in a second training stage the algorithm works in rounds. On each round the algorithm is provided with a new non-annotated sentence which it annotates, hopefully making the right decision for most of the words. Then the algorithm chooses subset of the words (or segments) to be annotated by humans. These words are the ones that algorithm estimates to be the hardest, or that their true label would resolve any ambiguity that is currently existing with the parsing of the input sentence.

Although such partial annotation task may be easier and faster to annotate, we realize that even partial annotation if not limited enough can require eventually similar effort as annotating the entire sentence. For example, if for a 25-words sentence annotation is requested for 5 words scattered over the entire sentence, providing this annotation may require the annotator to basically parse the entire sentence.

We thus further restrict the possible feedback requested from the annotator. Specifically, given a new sentence our algorithm outputs two possible annotations, or parse trees, $\hat{\boldsymbol{y}}_A$ and $\hat{\boldsymbol{y}}_B$, and asks for a single bit from the annotator, indicating whether parse A is better or parse B is better. We do not ask the annotator to parse the actual sentence, or decide what is the correct parse, but only to state which of the parses is quantitatively better. Formally, we say that parse $\hat{\boldsymbol{y}}_A$ is better if it contains more correct edges than $\hat{\boldsymbol{y}}_B$. The annotator is asked for a single bit, and thus must state one of the two parses, even if both parses are equally good. We denote this labeling paradigm as *binary*, as the annotator provides binary feedback.

The two parses our algorithms presents to the annotator are the highest ranking parse and the second highest ranking parse according to the current model. That is, the parse it would output for $\boldsymbol{x}$ and the best alternative. The feedback required from the annotator is only which of the two parses is better, the annotator does not explicitly indicate which of the edges are labeled correctly or incorrectly, and furthermore, the annotator does not provide any explicit information about the correct edge of any of the words.

In general, the two alternative parses presented to the annotator may be significantly different from each other; they may disagree on the edges of many words. In this case the task of deciding which of them is better may be as hard as annotating the entire sentence, and then comparing the resulting annotation to both alternatives. In practice, however, due to our choice of features (as functions of the two words) and model (linear), and since our algorithm chooses the two parse-trees ranked highest and second highest, the difference between the two alternatives is very small. In fact, we found empirically that, on average, in 95% of the sentences, they differ in the labeling of only a single word. That is, both $\hat{\boldsymbol{y}}_A$ and $\hat{\boldsymbol{y}}_B$ agree on all words, except some word x_i, for which the first alternative assigns to some word x_j and the second alternative assign to other word x_k. This is due to the fact that the score of the parses are *additive* in the edges. Therefore, the parse tree ranked second highest is obtained from the highest-ranked parse tree, where for a single word the edge is replaced, such that the difference between scores is minimized. For the remaining 5% of the sentences, replacing an edge as described causes a loop in the graph induced over words, and thus more than a single edge is modified. To minimize the potential labor of the annotator we simply ignore these cases, and present the annotator only two alternatives which are different in a single edge. We refer to this setting or scenario as *single*.

To conclude, given a new non-annotated sentence $\boldsymbol{x}$ the algorithm uses its current model $\boldsymbol{w}$ to output two annotations $\hat{\boldsymbol{y}}_A$ and $\hat{\boldsymbol{y}}_B$ which are different only on a *single* word and ask the annotator which is better. The annotator should decide to which of two possible words x_j and x_k to connect the word x_i in question. The annotator then feeds the algorithm a single bit, i.e. a *binary* labeling, which represents which alternative is better, and the algorithm updates its internal model $\boldsymbol{w}$. Although it may be the

```
Input data    A set of $n$ unlabeled sentences $\{\boldsymbol{x}_i\}_{i=1}^n$
Input parameters    Initial weight vector learned from fully annotated data $\boldsymbol{u}$; Number of Iterations over the unlabeled data $T$
Initialize    $\boldsymbol{w} \leftarrow \boldsymbol{u}$
For $t = 1, \ldots, T$
  • For $i = 1, \ldots, n$
      – Compute the two configurations $\hat{\boldsymbol{y}}_A$ and $\hat{\boldsymbol{y}}_B$ with highest scores of $\boldsymbol{x}_i$ using $\boldsymbol{w}$
      – Ask for feedback : $\hat{\boldsymbol{y}}_A$ vs. $\hat{\boldsymbol{y}}_B$
      – Get feedback $\beta \in \{+1, -1\}$ (or $\beta \in \{+1, 0, -1\}$ in Sec. 5)
      – Compute the value of $\alpha$ using the MIRA algorithm ( or just set $\alpha = 1$ for simplicity)
      – Update
          $\boldsymbol{w} + \alpha\beta \sum_{(i,j) \in (\hat{\boldsymbol{y}}_A / \hat{\boldsymbol{y}}_B \cup \hat{\boldsymbol{y}}_B / \hat{\boldsymbol{y}}_A)} (-1)^{[\![(i,j) \in \hat{\boldsymbol{y}}_B]\!]} \boldsymbol{f}(\boldsymbol{x}, i, j)$
Output:    Weight vector $\boldsymbol{w}$
```

Figure 1: The Light-Feedback learning algorithm

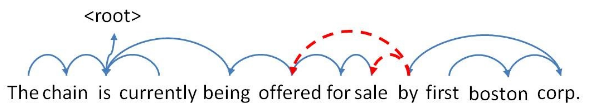
case that both alternatives are equally good (or bad), which occurs only when both assign the *wrong* word to x_i, that is not x_j nor x_k are the correct dependents of x_i, the annotator is still required to respond with one alternative, even though a wrong edge is recommended. Although this setting may induce noise, we consider it since a human annotator, that is asked to provide a quick light feedback, will tend to choose one of the two proposed options, the one that seems more reasonable, even if it is not correct. We refer to this combined setting of receiving a binary feedback only about a single word as *Binary-Single*. Below we discuss alternative models where the annotator may provide additional information, which we hypothesize, would be for the price of labor.

Finally, given the light-feedback β from the annotator, where $\beta = +1$ if the first parse $\hat{\boldsymbol{y}}_A$ is preferred over the second parse $\hat{\boldsymbol{y}}_B$, and $\beta = -1$ otherwise, we employ a single online update,

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\beta \left(\sum_{(i,j) \in \hat{\boldsymbol{y}}_A} \boldsymbol{f}(\boldsymbol{x}, i, j) - \sum_{(i,j) \in \hat{\boldsymbol{y}}_B} \boldsymbol{f}(\boldsymbol{x}, i, j) \right)$$

Pseudocode of the algorithm appears in Fig. 1.

From the last equation we note that the update depends only on the edges that are different between the two alternatives. This provides us the flexibility of what to show the annotator. One extreme is to provide the annotator with (almost) a full dependency parse tree, that both alternatives agree on, as well as the dilemma. This provides the annotator some context to assist of making a right decision and fast. The other extreme, is to provide the annotator only the edges for which the algorithm is not sure about, omitting any edges both alternatives agree on. This may remove labeling noise induced by erroneous edges both alternatives mistakenly agree on. Formally, these options are equivalent, and the decision which to use may even be dependent on the individual annotator.



<root>
The chain is currently being offered for sale by first boston corp.

Figure 2: Example of single edge feedback. The solid blue arrows describe the proposed parse and the two dashed red arrows are the requested light feedback.

An example of a light-feedback request is shown in Fig. 2. The sentence is 12 words long and the parser succeeded to assign correct edges for 11 words. It was uncertain whether there was a "sale by first boston corps" - having the edge "by→sale" (incorrect), or there was an "offer by first boston corps" - having the edge "by→offered" (correct). In this example, a human annotator can easily clarify the dilemma.

4 Evaluation

We evaluated the light feedback model using 14 languages: English (the Penn Tree Bank) and the remaining 13 were used in CoNLL 2006 shared task[1]. The number of training sentences in the training datasets is ranging is between about $1.5-57K$, with an average of about $14K$ sentences and $50K-700K$ words. The test sets contain an average of ~ 590 sentences and $\sim 10K$ words for all datasets. The average number of words per sentence vary from 6 in Chinese to 37 in Arabic.

[1]Arabic, Bulgarian, Chinese, Czech, Danish, Dutch, German, Japanese, Portuguese, Slovene, Spanish, Swedish and Turkish . See `http://nextens.uvt.nl/~conll/`

Experimental Setup For each of the languages we split the data into two parts of relative fraction of p and $1-p$ for $p = 10\%, 5\%$ and 1% and performed training in two stages. First, we used the smaller set to build a parser using standard supervised learning procedure. Specifically, we used MSTParser and ran the MIRA online learning algorithm for 5 iterations. This process yielded our initial parser. Second, the larger portion, which is the remaining of the training set, was used to improve the initial parser using the light feedback algorithm described above. Our algorithm iterates over the sentences of the larger subset and each sentence was parsed by the current parser (parameterized by $\boldsymbol{w}$) and asked for a preference between two specific parses for that sentence. Given this feedback, the algorithm updated its model and proceeded for the next sentence. The true parse of these sentences was only used to simulate light feedback and it was never provided to the algorithm. The performance of all the trained parsers was evaluated on a fixed test set. We performed five iterations of the larger subset during the light feedback training.

4.1 Results

The results of the light-feedback training after only a *single* iteration are given in the two left plots of Fig. 3. One plot shows the performance averaged over all languages, and second plot show the results for English. The black horizontal line shows the accuracy achieved by training the parser on the entire annotated data using the MIRA algorithm for 10 iterations. The predicted edge accuracy of the parser trained on the entire annotated dataset ranges from 77% on Turkish to 93% on Japanese, with an average of 85%. This is our *skyline*.

The blue bars in each plot shows the accuracy of a parser trained with only a fraction of the dataset - 10% (left group), via 5% (middle) to 1% (right group). As expected reducing the amount of training data causes degradation in performance, from an accuracy of about 76.3% (averaged over all languages) via 75% to 70.1% when training only with 1% of the data. These performance levels are our baselines, one per specific amount of fully annotated data and lightly annotated data.

The red bar in each pair, shows the contribution of a single training epoch with light-feedback on the performance. We see that training with light feedback improves the performance of the final parser. Most noticeably, is when using only 1% of the fully annotated data for initial training, and the remaining 99% of the training data with light feedback. The accuracy on test set improves from 70.1% to 75.6%, an absolute increase of 5.5%. These results are averaged over all languages, individual results for English are also shown. In most languages, including those not shown, these trends remain: when reducing the fraction of data used for fully supervised training the performance decreases, and light feedback improves it, most substantially for the smallest fraction of 1%.

We also evaluated the improvement in accuracy on the test set by allowing more than a single iteration over the larger fraction of the training set. The results are summarized in two right plots of Fig. 3, accuracy averaged over all languages (left), and for English (right). Each line refers to a different ratio of split between full supervised learning and light feedback learning - blue for 90%, green for 95% and red for 99%. The x-axis is the number of light feedback iterations, from zero up to five. The y-axis is the accuracy. In general more iterations translates to improvement in performance. For example, building a parser with only 1% of the training data yields 70.1% accuracy on the test set, a single iteration of light-feedback on the remaining 99% improves the performance to 75.6%, each of the next iterations improves the accuracy by about $1-2\%$ up to an accuracy of about 80%, which is only 5% lower than the skyline. We note again, that the skyline was obtained by using full feedback on the entire training set, while our parser used at most five bits of feedback per sentence from the annotator, one bit per iteration.

As noted above, on each sentence, and each iteration, our algorithm presents a parsing query or "dilemma": should word a be assigned to word b or word c. These queries are generated independently of the previous queries shown, and in fact the same query may be presented again in a later iteration although already shown in an early one. We thus added a memory storage of all queries to the algorithm. When a query is generated by the algorithm, it first checks if an annotation of it already exists in memory. If this is the case, then no query is issued to the annotator, and the algorithm simulates

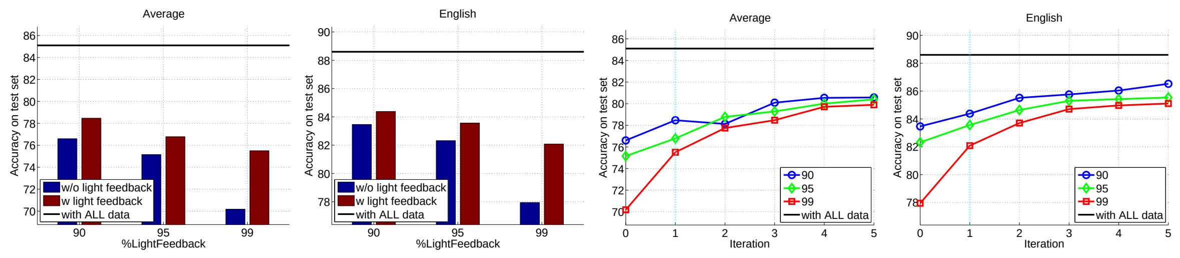


Figure 3: **Two left plots:** Evaluation in Binary-Single light feedback setting. Averaged accuracy over all languages (left) and for English. The horizontal black line shows the accuracy when training the parser on the entire annotated training data - "skyline". Each pair of bars shows the results for a parser trained with small amount of fully annotated data (left blue bar) and a parser that was then trained with a *single* iteration of light feedback on the remaining training data (right red bar). **Two right plots:** Evaluation of training with up to five iterations of binary-single light feedback. The plots show the average accuracy (left), and for English. Each line refers to a different ratio of split between full supervised learning and light feedback learning. The x-axis is the number of iterations of light feedback, from zero to five.

a query and response using the stored information.

The fraction of *new* queries, that were actually presented to the annotator, when light-training with 99% of the training set, is shown in the left panel of Fig. 4. Each line corresponds to one language. The languages are ordered in the legend according to the average number of words per sentence: from Chinese (6) to Arabic (37). Each point shows the fraction of new queries (from the total number of sentences with light-feedback) (y-axis) vs. the iteration (x-axis). Two trends are observed. First, in later iterations there are less and less new queries (or need for an actual interaction with the annotator). By definition, all queries during the first iteration are new, and the fraction of new queries after five iteration ranges from about 20% (Japanese and Chinese) to a bit less than 80% (Arabic).

The second trend is across the average number of words per sentence, the larger this number is, the more new queries there are in multiple iterations. For example, in Arabic (37 words per sentence) and Spanish (28) about 80% of the light-training sentences induce new queries in the fifth iteration, while in Chinese (6) and Japanese (8) only about 20%. As expected, longer sentences require, on average, more queries before getting their parse correctly.

We can also compare the performance improvement achieved by light feedbacks with the performance achieved by using the same amount of labeled-edges using fully annotated sentences in standard supervised training. The average sentence length across all languages is 18 words. Thus, receiving feedback regarding a single word in a sentence equals to about $1/18 \approx 5.5\%$ of the information provided by a fully annotated sentence. Therefore, we may view the light-feedback provided for 99% of the dataset as about equal to additional 5.5% of fully annotated data.

From the second plot from the right of Fig. 3, we see that by training with 1% of fully annotated data and a single iteration of light feedback over the remaining 99% of the data, the parser performance is 75.6% (square markers at $x = 1$), compared to 75% obtained by training with 5% of fully annotated data (diamond markers at $x = 0$). A second iteration of light feedback on 99% of the dataset can be viewed as *additional* $\lesssim 5\%$ of labeled data (accounting for repeating queries). After the second light feedback iteration, the parser performance is 77.8% (square markers at $x = 2$), compared to 76.3% achieved when training with 10% of fully annotated data (circle markers at $x = 0$). Similar relations can be observed for English in the right plot of Fig. 3. From these observations, we learn that on average, for about the same amount of labeled edges, light feedback learning gains equal, or even better, performance compared with fully labeled sentences.

5 Light Feedback Variants

Our current model is restrictive in two ways: first, the algorithm does not pass to the annotators examples for which the disagreements is larger than one word; and second, the annotator must prefer one of the two alternatives. Both restrictions were set to make the annotators' work easier. We now describe the results of experiments in which one or even both

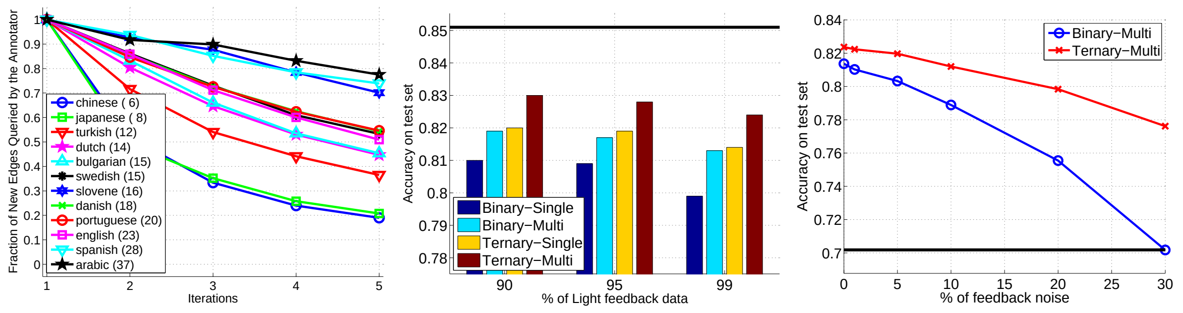


Figure 4: **Left:** the fraction of *new* queries presented to the annotator after each of the five iterations (x-axis) for all 14 languages, when light-training with 99% of the entire training data. **Middle:** comparison of the accuracy achieved using the four light feedback models using different fraction of the data for light feedback stage. The results are averaged over all the languages. **Right:** Effect of light-feedback noise on the accuracy of the trained model. Results are averaged over all languages for two light feedback settings, the ternary-multi and binary-multi. The plots show the performance measured on test set according the amount of feedback noise added. The black line is the baseline of the initial parser trained on 1% of annotated data.

restrictions are relaxed, which may make the work of the annotator harder, but as we shall see, improves performance.

Our first modification is to allow the algorithm to pass the annotator also queries on two alternatives $\hat{\boldsymbol{y}}_A$ and $\hat{\boldsymbol{y}}_B$ that differ on more than a single edge. As mentioned before, we found empirically that this arises in only $\sim$5% of the instances. In most cases the two alternatives differ in two edges, but in some cases the alternatives differ in up to five edges. Typically when the alternatives differ on more than a single edge, the words in question are close to each other in the sentence (in terms of word-distance) and are syntactically related to each other. For example, if changing the edge (i, j) to (i, k) forms a loop in the dependency graph then also another edge (k, l) must be changed to resolve the loop, so the two edges different between the alternatives are related. Nevertheless, even if the two alternatives are far from being similar, the annotator is still required to provide only a *binary* feedback, indicating a strict preference between the two alternatives. We refer to this model as *Binary-Multi*, for binary feedback and possibly multiple different edge between the alternatives.

Second, we enrich the number of possible responses of the annotator from two to three, giving the annotator the option to respond that the two alternatives $\hat{\boldsymbol{y}}_A$ and $\hat{\boldsymbol{y}}_B$ are equally good (or bad), and no one should be preferred by the other. In this case we set $\beta = 0$ in the algorithm of Fig. 1, and as can be seen in the pseudocode, this case does not modify the weight vector $\boldsymbol{w}$ associated with the parser. Such feedback will be received when both parses have the same number of errors. (We can also imagine a human annotator using the *equal* feedback to indicate "don't know"). For the common case of single edge difference between the two parses, this means that both proposed edges are incorrect. Since there are three possible responds we call this setting *ternary*. This setting can be combined with the previous one and thus we have in fact two new settings. The third setting is when only single edges are presented to the annotator, yet three possible responds are optional. We call this setting *Ternary-Single* . The fourth, is when the two alternatives may differ in more than a single edge *and* three possible responds are optional - *Ternary-Multi* setting.

The accuracy, averaged over all 14 languages, after 5 light feedback iterations, for all four settings is shown in the middle plate of Fig. 4. Each of the three groups summarizes the results for different split of the training set to full training and light-training: 90%, 95% and 99% (left to right; portion of light training). The horizontal black line shows the accuracy skyline (85% obtained by using all the training set in full supervised learning). Each bar in each group shows the results for one of the four settings: Binary-Single, Binary-Multi, Ternary-Single and Ternary-Multi. We focus our discussion in the 99% split. The averaged accuracy using Binary-Single feedback setting is about 80% (left bar). Relaxing the type of input to include alternatives that differ on more than one edge, improves accuracy by 1.4% (second bar from left). Slightly greater improvement is shown when relaxing the type of feedback, from binary to ternary (third bar from left). Finally, relaxing both constraints yields an improvement of additional 1% to an averaged accuracy of 82.5% which is only 2.5% lower than the skyline.

Moving to the other splits of $95, 90\%$ we observe that relaxing the feedback from binary to ternary improves the accuracy more than requiring to provide a preference of parses that differ on more than a single word.

6 Noisy Light Feedback

In the last section we discussed relaxations that requires slightly more effort from the annotator to gain higher test accuracy. The intent of the light feedback is to build a high-accuracy parser, yet faster and with less human effort compared with full supervised learning, or alternatively, allow collecting feedbacks from non-experts. We now evaluate the effect of light-feedback noise, which may be a consequence of asking the annotator to perform quick (and rough) light-feedback. We experiment with two settings in which the feedback of the annotator is either binary or ternary, in the *multi* settings, when 99% of the training-data is used for light-feedback. These settings refer to the second and fourth bar in the right group of the middle plate of Fig. 4.

We injected independent feedback errors to a fraction of ϵ of the queries, where ϵ is ranging between $0-30\%$. In the *Binary-Multi* setting, we flipped the binary preference with probability ϵ. For example, if $\hat{\boldsymbol{y}}_A$ is better than $\hat{\boldsymbol{y}}_B$ then with probability ϵ the light feedback was the other way around. In the *Ternary-Multi* setting we changed the correct feedback to one of the other two possible feedbacks with probability ϵ, the specific alternative chosen was chosen uniformly. E.g., if indeed $\hat{\boldsymbol{y}}_A$ is preferred over $\hat{\boldsymbol{y}}_B$, then with probability $\epsilon/2$ the feedback was that $\hat{\boldsymbol{y}}_B$ is preferred and with probability $\epsilon/2$ that both are equal.

The accuracy vs. noise level for both settings is presented in the right panel of Fig. 4. The black line shows the baseline performance after training an initial parser on 1% of annotated data. Performance of the parser trained using the *Binary-Multi* setting drops by only 1% from 81.4% to 80.4% at error rate of 5% and eventually as the feedback noise increase to 30% the performance drops to 70% - the performance level achieved by the initial trained model.

The accuracy of the parser trained in the richer *Ternary-Multi* setting suffers only 1% performance decrease at error rate of 10%, and eventually 5% decrease from 82.5% to 77.5% as the feedback noise increase to 30%, still a 7.5% improvement over the initial trained parser.

We hypothesize that learning with ternary feedback is more robust to noise, as in half of the noisy feedbacks when there is a strict preference between the alternatives, the effect of the noise is not to update the model and practically ignore the input. Clearly, this is preferable than the other outcome of the noise, that forces the algorithm to make the wrong update with respect to the true preference.

We also experimented with sentence depended noise by training a secondary parser on a subset of the training set, and emulating the feedback-bit using its output. Its averaged test error (=noise level) is 22%. Yet, the accuracy obtained by our algorithm with it is 77%, about the same as achieving with 30% random annotation noise. We hypothesize this is since the light-feedbacks are requested specifically on the edges harder to predict, where the error rate is higher than the 22% average error rate of the secondary parser.

7 Related work

Weak-supervision, semi-supervised and active learning (e.g. (Chapelle et al., 2006), (Tong and Koller, 2001)) are general approaches related to the light-feedback approach. These approaches build on access to a small set of labeled examples and a large set of unlabeled examples.

The work of Hall et al. (2011) is the most similar to the light feedback settings we propose. They apply an automatic implicit feedback approach for improving the performance of dependency parsers. The parser produces the k-best parse trees and an external system that uses these parse trees provides feedback as a score for each of the parses. In our work, we focus on minimal updates by both restricting the number of compared parses to two, and having them being almost identical (up to a single edge).

Hwa (1999) investigates training a phrase structure parser using partially labeled data in several settings. In one of the settings, a parser is first trained using a large fully labeled dataset from one domain and then adapted to another domain using partial labeling. The parts of the data that are labeled are selected in one of two approaches. In the first approach phrases are randomly selected to be annotated. In the second approach the phrases are selected according to their linguistic categories based on predefined rules. In both cases, the true phrases are provided. In our work, we train the initial parser on small subset of the data from the same domain. Additionally, the feedback queries are selected dynamically according to the edges estimated to be hardest for the parser. Finally, we request only limited feedback and the true parse is never provided directly.

Chang et al. (2007) use a set of domain specific rules as automatic implicit feedback for training information extraction system. For example, they use a set of 15 simple rules to specify the expected formats of fields to be extracted from advertisements. The light feedback regarding a prediction is the number of rules that are broken. That feedback is used to update the prediction model.

Baldridge and Osborne (2004) learns an HPSG parser using active learning to choose sentences to be annotated from a large unlabeled pool. Then, like our algorithm the annotator is presented with a proposed parse with several local alternatives sub-parse-trees. Yet, the annotator *manually* provides the correct parse, if it is not found within the proposed alternatives. Kristjansson et al. (2004) employ similar approach of combining active learning with corrective feedback for information extraction. Instances with lowest confidence using the current model are chosen to be annotated. Few alternative labels are shown to the user, yet again, the correct labeling is added *manually* if needed. The alternatives shown to the user are intended to reduce the effort of obtaining the right label, but eventually the algorithm receives the correct prediction. Our algorithm is passive about examples (and active only about subset of the labels), while their algorithm uses active learning to also choose examples. We plan to extend our work in this direction. Additionally, in these works, the feedback requests involve many alternatives and providing the true annotation, in oppose to the limited binary or ternary feedback. Yet our results show that despite of these limitations the trained parser achieved performance nor far from the performance of a parser training using the entire annotated dataset.

Finally, our setting is related to bandits (Cesa-Bianchi and Lugosi, 2006) where the feedback is extremely limited, a binary success-failure bit.

8 Summary

We showed in a series of experimental simulations that using light-feedback it is possible to train a dependency parser that achieves parsing performance not far from standard supervised training. Furthermore, very little amount of fully annotated data, even few tens of sentences, is sufficient for building an initial parser which can then be significantly improved using light-feedbacks.

While light-feedback training and standard supervised training with about the same number of total annotated edges may achieve close performance, we still view it as a possible alternative training framework. The reduction of the general annotation task into focused and small feedback requests, opens possibilities for receiving these feedbacks beyond expert labeling. In our ongoing work we study feedbacks from a large group of non-experts, and possibly even automatically. Additionally, we investigate methods for selecting light-feedback queries that are not necessarily derived from the highest scoring parse and the best alternative parse. For example, selecting queries that would be easy to answer by non-experts.

References

[Baldridge and Osborne2004] J. Baldridge and M. Osborne. 2004. Active learning and the total cost of annotation. In *EMNLP*.

[Cesa-Bianchi and Lugosi2006] N. Cesa-Bianchi and G. Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA.

[Chang et al.2007] Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

[Chapelle et al.2006] O. Chapelle, B. Schölkopf, and A. Zien, editors. 2006. *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

[Crammer et al.2006] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7:551–585.

[Hall et al.2011] Keith Hall, Ryan McDonald, Jason Katz-Brown, and Michael Ringgaard. 2011. Training dependency parsers by jointly optimizing multiple objectives. In *In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

[Hwa1999] Rebecca Hwa. 1999. Supervised grammar induction using training data with limited constituent information. *CoRR*, cs.CL/9905001.

[Kristjansson et al.2004] T. Kristjansson, A. Culotta, P. Viola, and A. McCallum. 2004. Interactive information extraction with constrained conditional random fields. In *AAAI*, pages 412–418.

[Marcus et al.1993] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Comput. Linguist.*, 19:313–330, June.

[McDonald et al.2005] R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT/EMNLP*.

[Tong and Koller2001] S. Tong and D. Koller. 2001. Support vector machine active learning with applications to text classification. In *JMLR*, pages 999–1006.