Verb Phrase Ellipsis Resolution Using Discriminative and Margin-Infused Algorithms

Kian Kenyon-Dean **Jackie Chi Kit Cheung** **Doina Precup**
School of Computer Science
McGill University
kian.kenyon-dean@mail.mcgill.ca, {jcheung,dprecup}@cs.mcgill.ca

Abstract

Verb Phrase Ellipsis (VPE) is an anaphoric construction in which a verb phrase has been elided. It occurs frequently in dialogue and informal conversational settings, but despite its evident impact on event coreference resolution and extraction, there has been relatively little work on computational methods for identifying and resolving VPE. Here, we present a novel approach to detecting and resolving VPE by using supervised discriminative machine learning techniques trained on features extracted from an automatically parsed, publicly available dataset. Our approach yields state-of-the-art results for VPE detection by improving F1 score by over 11%; additionally, we explore an approach to antecedent identification that uses the Margin-Infused-Relaxed-Algorithm, which shows promising results.

1 Introduction

Verb Phrase Ellipsis (VPE) is an anaphoric construction in which a verbal constituent has been omitted. In English, an instance of VPE consists of two parts: a **trigger**, typically an auxiliary or modal verb, that indicates the presence of a VPE; and an **antecedent**, which is the verb phrase to which the elided element resolves (Bos and Spenader, 2011; Dalrymple et al., 1991). For example, in the sentence, *"The government includes money spent on residential renovation; Dodge does not"*, the trigger *"does"* resolves to the antecedent *"includes money spent on residential renovation"*.

The ability to perform VPE resolution is important for tasks involving event extraction, especially in conversational genres such as informal dialogue where VPE occurs more frequently (Nielsen, 2005). Most current event extraction systems ignore VPE and derive some structured semantic representation by reading information from a shallow dependency parse of a sentence. Such an approach would not only miss many valid links between an elided verb and its arguments, it could also produce nonsensical extractions if applied directly on an auxiliary trigger. In the example above, a naive approach might produce an unhelpful semantic triple such as $(\textit{Dodge}, \textit{agent}, \textit{do})$.

There have been several previous empirical studies of VPE (Hardt, 1997; Nielsen, 2005; Bos and Spenader, 2011; Bos, 2012; Liu et al., 2016). Many previous approaches were restricted to solving specific subclasses of VPE (e.g., VPE triggered by *do* (Bos, 2012)), or have relied on simple heuristics for some or all of the steps in VPE resolution, such as by picking the most recent previous clause as the antecedent.

In this paper, we develop a VPE resolution pipeline which encompasses a broad class of VPEs (Figure 1), decomposed into the following two steps. In the *VPE detection* step, the goal is to determine whether or not a word triggers VPE. The second step, *antecedent identification*, requires selecting the clause containing the verbal antecedent, as well as determining the exact boundaries of the antecedent, which are often difficult to define.

Our contribution is to combine the rich linguistic analysis of earlier work with modern statistical approaches adapted to the structure of the VPE resolution problem. First, inspired by earlier work,

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1734–1743,
Austin, Texas, November 1-5, 2016.

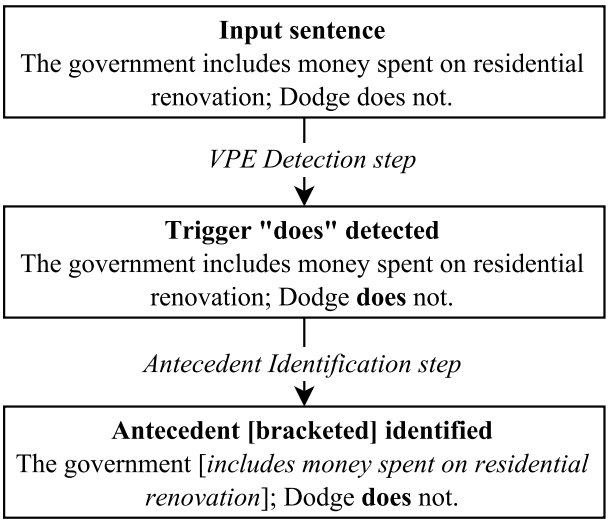

Figure 1: Example of the VPE resolution pipeline on an example found in WSJ file wsj_0036.

our system exploits linguistically informed features specific to VPE in addition to standard features such as lexical features or POS tags. Second, we adapt the Margin-Infused-Relaxed-Algorithm (MIRA) (Crammer et al., 2006), which has been popular in other tasks, such as machine translation (Watanabe et al., 2007) and parsing (McDonald et al., 2005), to antecedent identification. This algorithm admits a partial loss function which allows candidate solutions to overlap to a large degree. This makes it well suited to antecedent identification, as candidate antecedents can overlap greatly as well.

On *VPE detection*, we show that our approach significantly improves upon a deterministic rule-based baseline and outperforms the state-of-the-art system of Liu et al. (2016) by 11%, from 69.52% to 80.78%. For *antecedent identification* we present results that are competitive with the state-of-the-art (Liu et al., 2016). We also present state-of-the-art results with our end-to-end VPE resolution pipeline. Finally, we perform feature ablation experiments to analyze the impact of various categories of features.

2 Related Work

VPE has been the subject of much work in theoretical linguistics (Sag, 1976; Dalrymple et al., 1991, *inter alia*). VPE resolution could have a significant impact on related problems such as event coreference resolution (Lee et al., 2012; Bejan and Harabagiu, 2010; Liu et al., 2014) and event extraction (Ahn, 2006; Kim et al., 2009; Ritter et al., 2012). It has, however, received relatively little attention in the computational literature.

Hardt (1992) engaged in the first study of computational and algorithmic approaches for VPE detection and antecedent identification by using heuristic, linguistically motivated rules. Hardt (1997) extracted a dataset of 260 examples from the WSJ corpus by using an algorithm that exploited null elements in the PTB parse trees. Nielsen (2005) built a dataset that combined sections of the WSJ and BNC; he showed that the more informal settings captured in the BNC corpora show significantly more frequent occurrences of VPE, especially in dialogue excerpts from interviews and plays. Using this dataset, he created a full VPE pipeline from raw input text to a full resolution by replacing the trigger with the intended antecedent[1].

Bos and Spenader (2011) annotated the WSJ for occurrences of VPE. They found over 480 instances of VPE, and 67 instances of the similar phenomenon of *do-so* anaphora. Bos (2012) studied *do*-VPE by testing algorithmic approaches to VPE detection and antecedent identification that utilize Discourse Representation Theory.

Concurrently with the present work, Liu et al. (2016) explored various decompositions of VPE resolution into detection and antecedent identification subtasks, and they corrected the BNC annotations created by Nielsen (2005), which were difficult to use because they depended on a particular set of preprocessing tools. Our work follows a similar pipelined statistical approach. However, we explore an expanded set of linguistically motivated features and machine learning algorithms adapted for each subtask. Additionally, we consider all forms of VPE, including *to*-VPE, whereas Liu et al. only consider modal or light verbs (be, do, have) as candidates for triggering VPE. This represented about 7%

[1] e.g., the resolution of the example in Figure 1 would be "The government includes money spent on residential renovation; Dodge does not [include money spent on residential renovation]". We did not pursue this final step due to the lack of a complete dataset that explicitly depicts the correct grammatical resolution of the VPE.

Auxiliary Type	Example	Frequency
Do	does, done	214 (39%)
Be	is, were	108 (19%)
Have	has, had	44 (8%)
Modal	will, can	93 (17%)
To	to	29 (5%)
So	do so/same[3]	67 (12%)
TOTAL		554

Table 1: Auxiliary categories for VPE and their frequencies in all 25 sections of the WSJ.

of the dataset that they examined.

3 Approach and Data

We divide the problem into two separate tasks: VPE detection (Section 4), and antecedent identification (Section 5). Our experiments use the entire dataset presented in (Bos and Spenader, 2011). For preprocessing, we used CoreNLP (Manning et al., 2014) to automatically parse the raw text of WSJ for feature extraction. We also ran experiments using gold-standard parses; however, we did not find significant differences in our results[2]. Thus, we only report results on automatically generated parses.

We divide auxiliaries into the six different categories shown in Table 1, which will be relevant for our feature extraction and model training process, as we will describe. This division is motivated by the fact that different auxiliaries exhibit different behaviours (Bos and Spenader, 2011). The results we present on the different auxiliary categories (see Tables 2 and 4) are obtained from training a single classifier over the entire dataset and then testing on auxiliaries from each category, with the *ALL* result being the accuracy obtained over all of the test data.

[2]An anonymous reviewer recommended that further experiments could be performed by using the more informative NPs created with NML nodes (Vadas and Curran, 2007) on the gold-standard parsed WSJ.

[3]For example, "John will **go to the store** and Mary will *do the same/likewise/the opposite*". *Do X* anaphora and modals are not technically auxiliary verbs, as noted by the annotators of our dataset (Bos and Spenader, 2011), but for the purposes of this study we generalize them all as auxiliaries while simultaneously dividing them into their correct lexical categories.

4 VPE Detection

The task of VPE detection is structured as a binary classification problem. Given an auxiliary, a, we extract a feature vector f, which is used to predict whether or not the auxiliary is a trigger for VPE. In Figure 1, for example, there is only one auxiliary present, "does", and it is a trigger for VPE. In our experiments, we used a logistic regression classifier.

4.1 Feature Extraction

We created three different sets of features related to the auxiliary and its surrounding context.

Auxiliary. Auxiliary features describe the characteristics of the specific auxiliary, including the following:

- ◇ word identity of the auxiliary
- ◇ lemma of the auxiliary
- ◇ auxiliary type (as shown in Table 1)

Lexical. These features represent:

- ◇ the three words before and after the trigger
- ◇ their part-of-speech (POS) tags
- ◇ their POS bigrams

Syntactic. We devise these features to encode the relationship between the candidate auxiliary and its local syntactic context. These features were determined to be useful through heuristic analysis of VPE instances in a development set. The feature set includes the following binary indicator features (a = the auxiliary):

- ◇ a c-commands[4] a verb
- ◇ a c-commands a verb that comes after it
- ◇ a verb c-commands a
- ◇ a verb *locally*[5] c-commands a
- ◇ a locally c-commands a verb
- ◇ a is c-commanded by "than", "as", or "so"
- ◇ a is preceded by "than", "as", or "so"
- ◇ a is next to punctuation
- ◇ the word "to" precedes a
- ◇ a verb immediately follows a
- ◇ a is followed by "too" or "the same"

[4]A word *A* c-commands another word *B* if *A*'s nearest branching ancestor in the parse tree is an ancestor of B, following the definition of Carnie (2013). We use this term purely to define a syntactic relation between two points in a parse tree.

[5]A word *A* and word *B* share a local structure if they have the same closest S-node ancestor in the parse tree.

4.2 Baseline

As a baseline, we created a rule-based system inspired by Nielsen's (2005) approach to solving VPE detection. The baseline algorithm required significant experimental tuning on the development set because different linguistically hand-crafted rules were needed for each of the six trigger forms. For example, the following rule for *modals* achieved 80% F1-accuracy (see Table 2): "assume VPE is occurring if the modal does not c-command a verb that follows it". The other trigger forms, however, required several layers of linguistic rules. The rules for *be* and *have* triggers were the most difficult to formulate.

4.3 Experiments

We evaluate our models as usual using precision, recall and F1 metric for binary classification. The primary results we present in this section are obtained through 5-fold cross validation over all 25 sections of the automatically-parsed dataset. We use cross validation because the train-test split suggested by Bos and Spenader (2011) could result in highly varied results due to the small size of the dataset (see Table 1). Because the vast majority of auxiliaries do not trigger VPE, we over-sample the positive cases during training. Table 2 shows a comparison between the machine learning technique and a rule-based baseline for the six auxiliary forms. Table 3 shows results obtained from using the same train-test split used by Liu et al. (2016) in order to provide a direct comparison.

Results. Using a standard logistic regression classifier, we achieve an 11% improvement in accuracy over the baseline approach, as can be seen in Table 2. The rule-based approach was insufficient for *be* and *have* VPE, where logistic regression provides the largest improvements. Although we improve upon the baseline by 29%, the accuracy achieved for *be*-VPE is still low; this occurs mainly because: (i) *be* is the most commonly used auxiliary, so the number of negative examples is high compared to the number of positive examples; and, (ii) the analysis of the some of the false positives showed that there may have been genuine cases of VPE that were missed by the annotators of the dataset (Bos and Spenader, 2011). For example, this sentence (in file *wsj_2057*) was missed by the annotators (trigger in bold, antecedent italicized) "Some people tend to ignore that a 50-point move is *less in percentage terms* than it **was** when the stock market was lower."; here it is clear that **was** is a trigger for VPE.

Auxiliary	Baseline	ML	Change
Do	0.83	0.89	+0.06
Be	0.34	0.63	+0.29
Have	0.43	0.75	+0.32
Modal	0.80	0.86	+0.06
To	0.76	0.79	+0.03
So	0.67	0.86	+0.19
ALL	0.71	0.82	+0.11

Table 2: VPE detection results (baseline F1, Machine Learning F1, ML F1 improvement) obtained with 5-fold cross validation.

Test Set Results	P	R	F1
Liu et al. (2016)	**0.8022**	0.6134	0.6952
This work	0.7574	**0.8655**	**0.8078**

Table 3: Results (precision, recall, F1) for VPE detection using the train-test split proposed by Bos and Spenader (2011).

In Table 3, we compare our results to those achieved by Liu et al. (2016) when using WSJ sets 0-14 for training and sets 20-24 for testing. We improve on their overall accuracy by over 11%, due to the 25% improvement in recall achieved by our method. Our results show that oversampling the positive examples in the dataset and incorporating linguistically motivated syntactic features provide substantial gains for VPE detection. Additionally, we consider every instance of the word *to* as a potential trigger, while they do not - this lowers their recall because they miss every gold-standard instance of *to*-VPE. Thus, not only do we improve upon the state-of-the-art accuracy, but we also expand the scope of VPE-detection to include *to*-VPE without causing a significant decrease in accuracy.

5 Antecedent Identification

In this section we assume that we are given a trigger, from which we have to determine the correct antecedent; i.e., in the example in Figure 1, our task would be to identify "includes money spent on res-

idential renovation" as the correct antecedent. Our approach to this problem begins with generating a list of candidate antecedents. Next, we build a feature vector for each candidate by extracting features from the context surrounding the trigger and antecedent. Lastly, we use these features to learn a weight vector by using the Margin-Infused-Relaxed-Algorithm.

5.1 Candidate Generation

We generate a list of candidate antecedents by first extracting all VPs and ADJPs (and all contiguous combinations of their constituents) from the current sentence and the prior one. We then filter these candidates by predefining possible POS tags that an antecedent can start or end with according to the training set's gold standard antecedents. This method generates an average of 55 candidate antecedents per trigger, where triggers in longer sentences cause the creation of a larger number of candidate antecedents due to the larger number of VPs. This strategy accounts for 92% of the gold antecedents on the validation set by head match. We experimented with a less restrictive generation filter, but performance was not improved due to the much larger number of candidate antecedents.

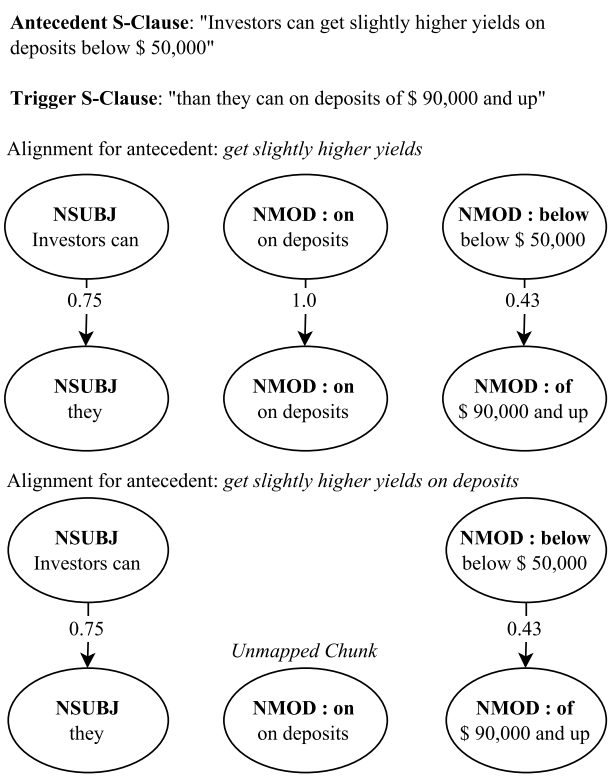

Figure 2: Alignment algorithm example with simplified dependencies.

5.2 Feature Extraction

We construct a feature vector representation for each candidate antecedent; in the example in Figure 1, for example, we would need feature vectors that differentiate between the two potential antecedents "includes money" and "includes money spent on residential renovation".

Alignment. This feature set results from an alignment algorithm that creates a mapping between the S-clause nearest to the trigger, S_t, and the S-clause nearest to the potential antecedent, S_a. The purpose of these features is to represent the parallelism (or lack thereof) between an antecedent's local vicinity with that of the trigger. The creation of this alignment algorithm was motivated by our intuition that the clause surrounding the trigger will have a parallel structure to that of the antecedent, and that an alignment between the two would best capture this parallelism. In the example sentence in Figure 2 (trigger in bold, antecedent italicized) "Investors can *get slightly higher yields* on deposits below $50,000 than they **can** on deposits of $90,000 and up" a simple observation of parallelism is that both the trigger and the correct antecedent are followed by the phrase "on deposits".

Formally, for each $S \in \{S_a, S_t\}$, we extract the dependencies in S as chunks of tokens, where each dependency chunk d_i contains all tokens between its governor and dependent (whichever comes first). Next, for each $d_i \in S_a$, if d_i contains any tokens that belong to the antecedent, delete those tokens. Similarly, for each $d_i \in S_t$, delete any token in d_i that belongs to T. We then perform a bipartite matching to align the $d_i \in S_t$ to the $d_j \in S_a$, where each edge's weight is determined by a scoring function $s(d_i, d_j)$. The scoring function we use considers the F1-similarity between the lemmas, POS-tags, and words shared between the two chunks, as well as whether or not the chunks share the same dependency name.

In the example in Figure 2 we can see that the correct antecedent, "*get slightly higher yields*", has a stronger alignment than the incorrect one, "*get slightly higher yields on deposits*". This occurs because we remove the candidate antecedent from its S-clause before creating the chunks; this leaves three nodes for the correct antecedent which map to the three nodes of the trigger's S-clause. However, this process only leaves two nodes for the incorrect candidate antecedent, thus causing one chunk to be unmapped, thus creating a weaker alignment.

We then use this mapping to generate a feature vector for the antecedent, which contains: the minimum, maximum, average, and standard deviation of the scores between chunks in the mapping; the number and percentage of unmapped chunks; the dependencies that have (and have not) been mapped to; the dependency pairs that were mapped together; and the minimum, maximum, average, and standard deviation of the cosine-similarity between the average word embedding of the words in a chunk between each d_i, d_j pair in the mapping.

NP Relation. These features compare the Noun Phrase (NP) closest to the antecedent to the NP closest to the trigger. This is motivated by an observation of many instances of VPE where it is often the case that the entity preceding the trigger is either repeated, similar, or corefers to the entity preceding the antecedent. The relationship between each *NP* is most significantly represented by features created with pre-trained *word2vec* word embeddings (Mikolov et al., 2013). For each NP, and for each word in the NP, we extract its pre-trained word embedding and then average them all together. We then use the cosine similarity between these two vectors as a feature.

Syntactic. Syntactic features are based on the relationship between the candidate antecedent's parse tree with that of the trigger. This feature set includes the following features, with the last three being influenced by Hardt's (1997) "preference factors" (a = candidate antecedent, t = trigger):

- ◇ if a's first word is an auxiliary
- ◇ if a's head (i.e., first main verb) is an auxiliary
- ◇ the POS tag of a's first and last words
- ◇ the frequency of each POS tag in the antecedent
- ◇ the frequency of each phrase (i.e., NP, VP, ADJP, etc.) in a's sentence and t's sentence
- ◇ if "than", "as", or "so" is between a and t
- ◇ if the word before a has the same POS-tag or lemma as t
- ◇ if a word in a c-commands a word in t
- ◇ if a's first or last word c-commands the trigger
- ◇ *Be-Do Form:* if the lemma of the token preceding a is *be* and the t's lemma is *do*
- ◇ *Recency:* distance between a and t and the distance between the t's nearest VP and a
- ◇ *Quotation:* if t is between quotation marks and similarly for a

Matching. This last feature set was influenced by the features described by Liu et al. (2016). We only use the "Match" features described by them; namely: whether the POS-tags, lemmas, or words in a two-token window before the start of the antecedent exactly match the two before the trigger; and whether the POS-tag, lemma, or word of the ith token before the antecedent equals that of the i-1th token before the trigger (for $i \in \{1, 2, 3\}$, where $i = 1$ considers the trigger itself).

5.3 Training Algorithm - MIRA

Since many potential antecedents share relatively similar characteristics, and since we have many features and few examples, we use the Margin-Infused-Relaxed-Algorithm (MIRA) in order to identify the most likely potential antecedent. MIRA maximizes the margin between the best candidate and the rest of the potential antecedents according to a loss function. It has been used for tasks with similar characteristics, such as statistical machine translation (Watanabe et al., 2007).

The training algorithm begins with a random initialization of the weight vector w. The training set contains triggers, each trigger's candidate antecedents, and their gold standard antecedents; it is reshuffled after each training epoch. We find the K highest-scoring potential antecedents, $a_1, \ldots, a_k$, according to the current weight value. A learning rate parameter determines how much we retain the new weight update with respect to the previous weight vector values.

MIRA defines the update step of the standard online training algorithm: it seeks to learn a weight

vector that, when multiplied with a feature vector f_i, gives the highest score to the antecedent that is most similar to the gold standard antecedent, a_*. This is posed as an optimization problem:

$$\begin{aligned} \underset{w_i}{\text{minimize}} \quad & \|w_i - w_{i-1}\| + C \sum_{k}^{K} \xi_k \\ \text{subject to} \quad & w_i \cdot a_* - w_i \cdot a_k + \xi_k \geq L(a_*, a_k), \\ & k = 1, \ldots, K \end{aligned} \tag{1}$$

Here, L is the loss function that controls the margin between candidates and the gold standard; it is defined as the evaluation metric proposed by Bos and Spenader (2011) (described in Section 5.5).

The ξ are slack variables and $C \geq 0$ is a hyper-parameter that controls the acceptable margin. This problem is solved by converting it to its Lagrange dual form[6].

5.4 Baseline Algorithm

The baseline we created was motivated by Bos's (2012) baseline algorithm: given a trigger, return as the antecedent the nearest VP that does not include the trigger. This is a naïve approach to antecedent identification because it does not consider the relationship between the context surrounding the antecedent and the context surrounding the trigger.

5.5 Experiments

We evaluate our results following the proposed metrics of Bos and Spenader (2011), as do Liu et al. (2016). Accuracy for antecedent identification is computed according to n = the number of correctly identified tokens between the candidate antecedent and the gold standard antecedent. Precision is n divided by the length of the candidate antecedent, recall is n divided by the length of the correct antecedent, and accuracy is the harmonic mean of precision and recall. For MIRA, final results are determined by choosing the weight vector that achieved the best performance on a validation set that is split off from part of the training set, as calculated after each update step.

[6]In this study, the dual form was implemented by hand using Gurobi's python API (Gurobi Optimization Inc., 2015).

Auxiliary	Baseline	MIRA	Change
do	0.42	0.71	+0.29
be	0.37	0.63	+0.26
modal	0.42	0.67	+0.25
so	0.15	0.53	+0.38
have	0.39	0.61	+0.22
to	0.03	0.58	+0.55
ALL	0.36	0.65	+0.29

Table 4: Results (baseline accuracy, MIRA accuracy, accuracy improvement) for antecedent identification; obtained with 5-fold cross validation.

End-to-end Results	P	R	F1
Liu et al. (2016)	**0.5482**	0.4192	0.4751
This work	0.4871	**0.5567**	**0.5196**

Table 5: End-to-end results (precision, recall, F1) using the train-test split proposed by Bos and Spenader (2011).

MIRA has several hyper-parameters that were tuned through a grid search over the validation set. The most crucial parameters were the learning rate α, and C, while the value of K did not cause significant changes in accuracy.

Results. In Table 4, we see that MIRA improves upon the baseline with a 29% increase in overall accuracy. MIRA provides significant gains for each form of VPE, although there is room for improvement, especially when identifying the antecedents of *do-so* triggers.

Liu et al. (2016) achieve an accuracy of 65.20% with their joint resolution model for antecedent identification when using the train-test split proposed by Bos and Spenader (2011); our model achieves 62.20% accuracy. However, their experimental design was slightly different than ours — they only considered antecedents of triggers detected by their oracle trigger detection method, while we use all gold-standard triggers, meaning our results are not directly comparable to theirs. Our cross validated results (65.18% accuracy) paint a better picture of the quality of our model because the small size of the dataset (554 samples) can cause highly varied results.

Excluded	P	R	F1
Auxiliary	0.7982	0.7611	0.7781
Lexical	0.6937	0.8408	0.7582
Syntactic	0.7404	0.7330	0.7343
NONE	0.8242	0.8120	0.8170

Table 6: Feature ablation results (feature set excluded, precision, recall, F1) on VPE detection; obtained with 5-fold cross validation.

Features Excluded	Accuracy
Alignment	0.6511
NP Relation	0.6428
Syntactic	0.5495
Matching	0.6504
NONE	0.6518

Table 7: Feature ablation results (feature set excluded, precision, recall, F1) on antecedent identification; obtained with 5-fold cross validation.

In Table 5 we present end-to-end results obtained from our system when using the triggers detected by our VPE detection model (see Section 4). We compare these results to the end-to-end results of the best model of Liu et al. (2016). Following Liu et al., we assign partial credit during end-to-end evaluation in the following way: for each correctly detected (true positive) trigger, the Bos and Spenader (2011) antecedent evaluation score between the trigger's predicted antecedent and its gold antecedent is used (as opposed to a value of 1). As can be seen from Table 5, we trade about 6 points of precision for 14 points of recall, thus improving state-of-the-art end-to-end accuracy from 47.51% to 51.96%.

6 Feature Ablation Studies

We performed feature ablation experiments in order to determine the impact that the different feature sets had on performance.

Trigger Detection. In Table 6 we can see that the syntactic features were essential for obtaining the best results, as can be seen by the 8.3% improvement, from 73.4% to 81.7%, obtained from including these features. This shows that notions from theoretical linguistics can prove to be invaluable when approaching the problem of VPE detection and that extracting these features in related problems may improve performance.

Antecedent Identification. Table 7 presents the results from a feature ablation study on antecedent identification. The most striking observation is that the alignment features do not add any significant improvement in the results. This is either because there simply is not an inherent parallelism between the trigger site and the antecedent site, or because the other features represent the parallelism adequately without necessitating the addition of the alignment features. The heuristic syntactic features provide a large (10%) accuracy improvement when included. These results show that a dependency-based alignment approach to feature extraction does not represent the parallelism between the trigger and antecedent as well as features based on the lexical and syntactic properties of the two.

7 Conclusion and Future Work

We presented an approach for the tasks of Verb Phrase Ellipsis detection and antecedent identification that leverages features informed both by theoretical linguistics and NLP, and employs machine learning methods to build VPE detection and antecedent identification tools using these features. Our results show the importance of distinguishing VPE triggers from each other, and highlight the importance of using the notion of c-command for both tasks.

For VPE detection, we improve upon the accuracy of the state-of-the-art system by over 11%, from 69.52% to 80.78%. For antecedent identification, our results significantly improve upon a baseline algorithm and we present results that are competitive with the state-of-the-art, as well as state-of-the-art results for an end-to-end system. We also expand the scope of previous state-of-the-art by including the detection and resolution of *to*-VPE, thus building a system that encompasses the entirety of the Bos and Spenader (2011) VPE dataset.

In future work, we would like to further inves-

tigate other margin-based optimizations similar to MIRA, but perhaps even more resilient to overfitting. We also seek to improve the antecedent identification approach by extracting stronger features.

Acknowledgments

This work was funded by McGill University and the Natural Sciences and Engineering Research Council of Canada via a Summer Undergraduate Research Project award granted to the first author. We thank the anonymous reviewers for their helpful suggestions, and we thank Nielsen, Hector Liu, and Edgar Gonzàlez for their clarifying remarks over email.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics.

Cosmin Adrian Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1412–1422. Association for Computational Linguistics.

Johan Bos and Jennifer Spenader. 2011. An annotated corpus for the analysis of VP ellipsis. *Language Resources and Evaluation*, 45(4):463–494.

Johan Bos. 2012. Robust VP ellipsis resolution in DR theory. In Staffan Larsson and Lars Borin, editors, *From Quantification to Conversation*, volume 19 of *Tributes*, pages 145–159. College Publications.

Andrew Carnie. 2013. *Syntax: A generative introduction*. John Wiley & Sons.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

Mary Dalrymple, Stuart M Shieber, and Fernando CN Pereira. 1991. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14(4):399–452.

Gurobi Optimization Inc. 2015. Gurobi optimizer reference manual.

Daniel Hardt. 1992. An algorithm for VP ellipsis. In *Proceedings of the 30th Annual Meeting on Association for Computational Linguistics*, pages 9–14. Association for Computational Linguistics.

Daniel Hardt. 1997. An empirical approach to VP ellipsis. *Computational Linguistics*, 23(4):525–541.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP '09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9. Association for Computational Linguistics.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500. Association for Computational Linguistics.

Zhengzhong Liu, Jun Araki, Eduard H Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *LREC*, pages 4539–4544.

Zhengzhong Liu, Edgar Gonzalez, and Dan Gillick. 2016. Exploring the steps of verb phrase ellipsis. In *Proceedings of the Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2016), co-located with NAACL 2016*, pages 32–40.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Leif Arda Nielsen. 2005. *A Corpus-Based Study of Verb Phrase Ellipsis Identification and Resolution*. Ph.D. thesis, King's College London.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Ivan A Sag. 1976. *Deletion and logical form.* Ph.D. thesis, Massachusetts Institute of Technology.

David Vadas and James Curran. 2007. Adding noun phrase structure to the penn treebank. In *Annual Meeting - Association for Computational Linguistics*, volume 45, page 240.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773.