Automatically Tagging Constructions of Causation and Their Slot-Fillers

Jesse Dunietz
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
jdunietz@cs.cmu.edu

Lori Levin and **Jaime Carbonell**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{lsl,jgc}@cs.cmu.edu

Abstract

This paper explores extending shallow semantic parsing beyond lexical-unit triggers, using causal relations as a test case. Semantic parsing becomes difficult in the face of the wide variety of linguistic realizations that causation can take on. We therefore base our approach on the concept of CONSTRUCTIONS from the linguistic paradigm known as CONSTRUCTION GRAMMAR (CxG). In CxG, a construction is a form/function pairing that can rely on arbitrary linguistic and semantic features. Rather than codifying all aspects of each construction's form, as some attempts to employ CxG in NLP have done, we propose methods that offload that problem to machine learning. We describe two supervised approaches for tagging causal constructions and their arguments. Both approaches combine automatically induced pattern-matching rules with statistical classifiers that learn the subtler parameters of the constructions. Our results show that these approaches are promising: they significantly outperform naïve baselines for both construction recognition and cause and effect head matches.

1 Introduction

Historically, shallow semantic parsing has focused on tagging predicates expressed by individual lexical units. While this paradigm has been fruitful, tying meaning to lexical units excludes some essential semantic relationships that cannot be captured in such a representation.

One domain that highlights the problem is causal relations. Causation can be expressed in a tremendous variety of linguistic forms (Wolff et al., 2005). As exemplified in Table 1, possibilities include verbs (1, 2), prepositions/conjunctions (3, 4, 5), adjectives (6), and much more complex expressions. Some of these trickier cases can be handled as idiomatic multiword expressions (MWEs; 7). Others (8, 9), however, are more structured than typical MWEs: they depend on particular configurations of syntactic relations and slot-fillers, placing them closer to the grammatical end of the continuum of lexicon and grammar.

1. THIS BILL **promotes** *consolidation and cooperation among regulatory agencies*.
2. SUCH SWELLING can **impede** *breathing*.
3. WE DON'T HAVE MUCH TIME, **so** *let's move quickly*.
4. *She's mad* **because** I HID THE CAR KEYS.
5. *He died* **from** A BLOCKED ARTERY.
6. *Making money* is **contingent on** FINDING A GOOD-PAYING JOB.
7. THIS DECISION **opens the way for** *much broader application of the law*.
8. **For** *market discipline* **to** *work*, BANKS CANNOT EXPECT TO BE BAILED OUT.
9. Judy's comments were **SO** OFFENSIVE **that** *I left*.

Table 1: Examples of causal language, reflecting the annotation scheme described in §2.1 (with **connectives** in bold, CAUSES in small caps, and *effects* in italics).

This diversity presents a problem for most semantic parsers, which inherit the restrictions of the representational schemes they are based on. Many semantic annotation schemes limit themselves to the argument structures of particular word classes. For example, the Penn Discourse Treebank (PDTB;

Transactions of the Association for Computational Linguistics, vol. 5, pp. 117–133, 2017. Action Editor: Christopher Potts.
Submission batch: 9/2016; Revision batch: 11/2016; Published 6/2017.

Prasad et al., 2008) includes only conjunctions and adverbials as connectives,[1] and PropBank (Palmer et al., 2005) and VerbNet (Schuler, 2005) focus on verb arguments. FrameNet (Baker et al., 1998; Fillmore, 2012) is less restrictive, allowing many parts of speech as triggers.

Most importantly, though, all these representations share the fundamental simplifying assumption that the basic linguistic carrier of meaning is the lexical unit. Some (e.g., PDTB and FrameNet) allow MWEs as lexical units, and much work has been done on detecting and interpreting MWEs (see Baldwin and Kim, 2010). But even these schemes overlook essential linguistic elements that encode meanings. In example 9, for instance, a lexical unit approach would have to treat *so* as encoding the causal relationship, when in fact *so* merely intensifies the adjective; it is the combination of *so* and the finite clausal complement that indicates causality.

A more general approach can be found in the principles of CONSTRUCTION GRAMMAR (CxG; Fillmore et al., 1988; Goldberg, 1995). CxG posits that the fundamental units of language are CONSTRUCTIONS – pairings of meanings with arbitrarily complex linguistic forms. These forms are often productive, consisting of some fixed elements combined with some open slots for semantic arguments. The form/meaning pairings can be as simple as those in traditional lexical semantics. The verb *push*, for instance, is paired with the meaning *force to move*, and the verb takes two linguistic arguments (subject and object) corresponding to the two semantic arguments (pusher and pushee). But in CxG, the meaning-bearing forms can be much more complex: *so X that Y* is a single construction, paired with the meaning *X to an extreme that causes Y*.

The CxG paradigm can anchor semantic interpretations to any constellation of surface forms, making it potentially invaluable for computational semantics. Even as it has grown in prominence in linguistics, however, CxG has received relatively little attention in NLP. This is partly because the usual approach to operationalizing CxG is to rebuild the entire NLP pipeline to be "constructions all the way down" – to explicitly model the interactions and inheritance relationships between constructions that produce the final utterance and meaning.

Here, we take a different approach. Instead of "constructions all the way down," we propose a "CONSTRUCTIONS ON TOP" methodology: we use a conventional NLP pipeline for POS tagging, parsing, and so on, but add a layer for constructional phenomena that directly carry meaning. Rather than specifying by hand the constraints and properties that characterize each construction, we allow machine learning algorithms to learn these characteristics.

Causal relations present an ideal testbed for this approach. As noted above, causal relations are realized in extremely diverse ways, demanding an operationalized concept of constructions. Recognizing causal relations also requires a combination of linguistic analysis and broader world knowledge. Additionally, causal relations are ubiquitous, both in our thinking and in our language (see, e.g., Conrath et al., 2014). Recognizing these relations is thus invaluable for many semantics-oriented applications, including textual entailment and question answering (especially for "why" questions). They are especially helpful for domain-specific applications such as finance, politics, and biology (see, e.g., Berant et al., 2014), where extracting cause and effect relationships can help drive decision-making. More general applications like machine translation and summarization, which ought to preserve stated causal relationships, can also benefit.

In the remainder of this paper, we suggest two related approaches for tagging causal constructions and their arguments. We first review an annotation scheme for causal language and present a new corpus annotated using that scheme (§2). We then define the task of tagging causal language, casting it as a construction recognition problem (§3). Because it is so hard to identify the relevant components of a construction (tenses, grammatical relations, etc.), the scheme and task do not explicitly include all of these elements. We instead tag the words that participate in a causal construction as a proxy for that construction. We leave it to annotators (when humans are annotating) or machine learning (during automated tagging) to assess when the full constellation of constructional elements is present.

Next, we present Causeway-L and Causeway-S, two versions of a pipeline for performing this task,

[1] PDTB does include a catch-all AltLex category that captures some additional constructions. However, these phrases are very unpredictable, as they include many words beyond the linguistic triggers. They are also restricted to relations **between** sentences.

and compare the two approaches. Both approaches use automatically induced patterns, either syntactic (§4.1) or lexical (§4.2), to find possible lexical triggers of causal constructions, as well as their likely arguments. They then apply a mix of construction-specific classifiers and construction-independent classifiers to determine when causal constructions are truly present. We report on three sets of experiments (§5) assessing the two systems' performance, the impacts of various design features, and the effects of parsing errors. The results indicate the viability of the approach, and point to further work needed to improve construction recognition (§6).

2 Causal Language Annotation Scheme and Corpus

Causation is a slippery notion (see Schaffer, 2014), so the parameters of annotating causal language require careful definition. We follow the annotation scheme of Dunietz et al. (2015), which we now briefly review.

2.1 Causal Language Annotation Scheme

The scheme of Dunietz et al. (2015) focuses specifically on **causal language** – language used to appeal to psychological notions of cause and effect. It is not concerned with what causal relationships hold in the real world; rather, it represents what causal relationships are asserted by the text. For example, *cancer causes smoking* states a false causation, but it would nonetheless be annotated. On the other hand, *the bacon pizza is delicious* would not be annotated, even though bacon may in fact cause deliciousness, because the causal relationship is not stated.

The scheme defines causal language as any construction which presents one event, state, action, or entity as promoting or hindering another, and which includes at least one lexical trigger. For each instance of causal language, up to three spans are annotated:

- **The causal connective** (required) – the lexical items in the construction signaling the causal relationship (e.g., *because of*). The connective annotation includes all words whose lemmas appear in every instance of the construction. This excludes elements that can be absent, such as most copulas, or classes of interchangeable words, such as determiners, whose lemmas can vary between instances.
- **The cause** – generally a full clause or phrase expressing an event or state of affairs (e.g., *I attended because* ***Joan was the honoree***). When an actor – but no action – is presented as the cause (e.g., ***I*** *prevented a fire.*), the actor is annotated as the cause.
- **The effect** – also generally an event or state of affairs, expressed as a complete clause or phrase (e.g., ***I attended*** *because of Joan.*).

The cause and effect spans may be absent, e.g., in a passive or infinitive. Several examples, with connectives, causes, and effects annotated, are shown in Table 1.

The scheme permits the connective to be discontinuous and arbitrarily complex (e.g., *a necessary condition of* or *if __ is to __*). Annotators together established an informal "constructicon" to guide their decisions about what word patterns should be considered connectives. Note that the connective is not synonymous with the construction itself; rather, it is a lexical indicator of the presence of the construction. As noted above, we take the construction proper to include the relevant grammatical relations, constraints on argument type, etc.

To address causal language as independently as possible from other phenomena, and to circumscribe the scope of the annotations, several types of relationships are excluded:

- **Causal relationships with no lexical trigger** – e.g., *John went home early; he felt ill.*
- **Connectives that incorporate a means or result** – this includes lexical causatives such as *kill* (cause to die) and *convince* (cause via persuasion). Only connectives denoting pure causation (e.g., *cause*, *prevent*, *because*) are annotated.
- **Connectives that assert an unspecified causal relationship** – e.g., *the earthquakes have been linked to fracking.*
- **Temporal language** – e.g., *after I drank some water, I felt much better*. (Relations like temporal order are often repurposed to express causality. We are currently developing an enhanced annotation scheme and corpus that accounts for such cases.)

Additionally, for practical reasons, arguments are only annotated when they appear within the same sentence as the connective.

The scheme labels four different types of causation: CONSEQUENCE, MOTIVATION, PURPOSE, and INFERENCE. It also distinguishes positive causation (FACILITATE) from negative causation (INHIBIT). Our algorithms do not currently make these distinctions, so we do not delve into them here.

Of course, this scheme does not supply everything a full natural language understanding pipeline would need regarding causal relationships. The scheme does not unpack the argument spans into a richer semantic representation. It also does not cover predicates that imply causation, such as *kill* or *convince*. Instead, it follows in the tradition of shallow semantic parsing, imposing a canonical representation for certain predicates that abstracts away from the language used to convey them. The shallow semantic parsing paradigm enables many applications that only require relevant text spans. It is also the first step towards a full semantic interpretation that includes interpretations of the arguments.

2.2 The BECAUSE Annotated Corpus

Based on the data and annotation scheme from Dunietz et al. (2015), we developed the **B**ank of **E**ffects and **Cau**ses **S**tated **E**xplicitly (BECAUSE), which was used for all experiments below. It consists of three sets of exhaustively annotated documents:

- 59 randomly selected articles from the year 2007 in the Washington section of the New York Times corpus (Sandhaus, 2008)
- 47 documents randomly selected from sections 2–23 of the Penn Treebank[2] (Marcus et al., 1994)
- 679 sentences[3] transcribed from Congress' Dodd-Frank hearings, taken from the NLP Unshared Task in PoliInformatics 2014 (Smith et al., 2014)

The corpus contains a total of 4161 sentences, among which are 1099 labeled instances of causal language. 1004 of these, or 91%, include both cause and effect arguments.

Many of these documents are the same ones that were annotated in Dunietz et al. (2015), with minor corrections. About 75% of the data is new, but annotated by the same annotators. The scheme's inter-annotator agreement metrics are reproduced in Table 2.

[2]We excluded WSJ documents that were either earnings reports or corporate leadership/structure announcements, as both tended to be merely short lists of names/numbers.

[3]The remaining sentences and documents were not annotated due to constraints on available annotation effort.

	Partial overlap: Allowed	Partial overlap: Excluded
Connectives (F_1)	0.78	0.70
Degrees (κ)	1.0	1.0
Causation types (κ)	0.82	0.80
Argument spans (F_1)	0.96	0.86
Argument labels (κ)	0.98	0.97

Table 2: Inter-annotator agreement results for the BECAUSE corpus. The difference between the two columns is that for the left column, we counted two annotation spans as a match if at least a quarter of the larger one overlapped with the smaller; for the right column, we required an exact match. κ scores indicate Cohen's kappa. Each κ score was calculated only for spans that agreed (e.g., degrees were only compared for matching connective spans).

Corpus	Connective types	Connective tokens
PDTB	8.9%	34.4%
Mirza and Tonelli (2014)	29.3%	47.5%
FrameNet	69.4%	66.9%

Table 3: Percentages of the causal connectives in BECAUSE that would be partially or fully annotatable under other annotation schemes. Connectives were grouped into types by the sequence of connective lemmas.

Many of the causal constructions in BECAUSE would be harder to annotate in other schemes, as shown in Table 3. We computed these statistics by looking up each connective in the other schemes' lexica. FrameNet captures many more connectives than the others, but it often represents them in frames that are not linked to causality, making comparison difficult.

3 The Causal Language Tagging Task

We define the task of tagging causal language to be reproducing a subset of the annotations of the BECAUSE annotation scheme. We split this task into two parts:

1. **Connective discovery**, in which the spans of causal connectives are annotated. A connective

span may be any set of tokens from the sentence. This can be thought of as recognizing instantiations of causal constructions.

2. **Argument identification** (or argument ID), in which cause and effect spans are identified for each causal connective. This can be thought of as identifying the causal construction's slot-fillers.

We assume as input a set of sentences, each with POS tags, lemmas, NER tags, and a syntactic parse in the Universal Dependencies (UD; Nivre et al., 2016) scheme, all obtained from version 3.5.2 of the Stanford parser (Klein and Manning, 2003).[4]

This task is defined in terms of text spans. Still, to achieve a high score on it, a tagger must respond to the meaning of the construction and arguments in context, just as annotators do. This may be achieved by analyzing indirect cues that correlate with meaning, such as lexical information, dependency labels, and tense/aspect/modality information.

Compared to the annotation scheme, the task is limited in two important ways: first, we do not distinguish between types or degrees of causation; and second, we only tag instances where both the cause and the effect are present. (Even for connective discovery, we only evaluate on instances where both arguments are present, and our algorithms check for spans or tokens that at least could be arguments.) We leave addressing both limitations to future work.

Nonetheless, this task is more difficult than it may appear. Two of the reasons for this are familiar issues in NLP. First, there is a surprisingly long tail of causal constructions (as we finished annotating, we would still encounter a new construction every 2–3 documents). Second, recognizing these constructions inherits all the difficulties of word sense disambiguation; every one of the connectives in Table 1 except examples 6 and 7 has a non-causal meaning. (4 can be used in a discourse sense – roughly, *I'm saying this because.*...) The third reason arises from the complexity of causal constructions. Because we allow arbitrarily complex connectives, the space of possible triggers is all subsets of words in the sentence. Thus, the task demands an approach that can trim this space down to a manageable size.

[4]We use the non-collapsed enhanced dependency representation. We could have selected a parser that produces both syntactic and semantic structures, such as the English Resource Grammar (ERG; Copestake and Flickinger, 2000) or another HPSG variant. Though these parsers can produce impressively sophisticated analyses, we elected to use dependency parsers because they proved significantly more robust; there were many sentences in our corpus that we could not parse with ERG. However, incorporating semantic information from such a system when it is available would be an interesting extension for future work.

Another possible input would have been semantic role labeling (SRL) tags. SRL tags could not form the basis of our system the way syntactic relations can, because they only apply to limited classes of words (primarily verbs). Also, by examining syntactic relations and undoing passives (see §), we get most of the information SRL would provide. Still, we may include SRL tags as classification features in the future.

4 Causeway: Causal Construction Tagging Methods

Causeway is a system that performs this causal language tagging task. We implemented two versions of Causeway: Causeway-S, based on syntactic patterns, and Causeway-L, based on lexical patterns. (These are both simple techniques; see §8 for some more complex possibilities we are considering for future work.) Each technique is implemented as a pipeline with four stages:

1. **Pattern-based tentative connective discovery**. Both techniques extract lexical or lexico-syntactic patterns for connectives from the training data. These patterns are then matched against each test sentence to find tokens that **may** be participating in a causal construction.
2. **Argument identification**, which marks the cause and effect spans.
3. **A statistical filter** to remove false matches.
4. **A constraint-based filter** to remove redundant connectives. Smaller connectives like *to* (which is causal in sentences like *I left to get lunch*)[5] are usually spurious when a larger connective includes the same word, like *cause* X *to* Y. When a larger and a smaller connective both make it through Stage 3 together, we remove the smaller one.

Because argument ID is done before filtering, the arguments output by Stage 2 do not quite represent the cause and effect arguments of a causal instance.

[5]Following Schneider et al. (2015), BECAUSE considers the "in order to" usage of the infinitive *to* to carry lexical meaning beyond just marking an infinitival clause.

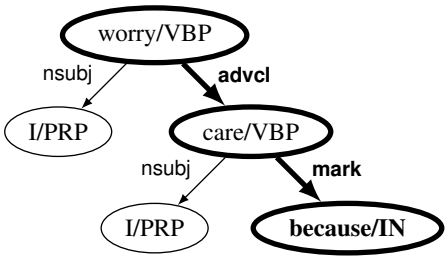

Figure 1: A UD parse for the sentence *I worry because I care*, with the tree fragment corresponding to the *because* construction bolded. Bolded nodes with unbolded text indicate the slots in the construction. (*Care* is a dependent of *worry* in keeping with the UD design philosophy, which maximizes dependencies between content words.)

Rather, they represent what the cause and effect spans would be **if** the connective is indeed causal. (For the same reason, even connective discovery is not complete until the end of the pipeline.) Of course, we lack gold-standard arguments for false-positive connectives. We therefore train argument ID only on instances whose connectives are correct.

We now describe each of the two versions of this pipeline in turn, focusing on their differing first and second stages. We also elaborate on the design of the classifier. Throughout, we take the head of a span to be the token that is highest in the dependency tree. For tokens that are equally high (e.g., if the parser erroneously splits a phrase into two sister subtrees), we prefer verbs or post-copular nouns over other nouns, and nouns over other parts of speech.

4.1 Causeway-S: Syntax-Based Tagging

The syntax-based approach relies on a simple intuition: each causal construction corresponds, at least in part, to a fragment of a dependency tree, where several nodes' lemmas and POS tags are fixed (see Figure 1). Accordingly, the first stage of Causeway-S induces lexico-syntactic patterns from the training data. At test time, it matches the patterns against new dependency trees to identify possible connectives and the putative heads of their cause and effect arguments. The second stage then expands these heads into complete argument spans.

TRegex Pattern Matching

For pattern matching, we use TRegex (Levy and Andrew, 2006), a grep-inspired utility for matching patterns against syntax trees. During training, the system examines each causal language instance, generating a TRegex pattern that will match tree fragments with the same connective and argument structure. In the example from Figure 1, the generated pattern would match any tree meeting three conditions:[6]

- some token t_1 has a child t_2 via a dependency labeled advcl
- t_2 has a child t_3 via a dependency labeled mark
- t_3 has the lemma *because* and a POS tag of IN

At test time, TRegex matches these extracted patterns against the test sentences. Continuing with the same example, we would recover t_1 as the effect head, t_2 as the cause head, and $\{t_3\}$ as the connective.

TRegex is designed for phrase-structure trees, so we transform each dependency tree into a PTB-like parenthetical notation (see Levin et al., 2014).

Patterns involving verbs vary systematically: the verbs can become passives or verbal modifiers (e.g., *the disaster averted last week*), which changes the UD dependency relationships. To generalize across these, we crafted a set of scripts for TSurgeon (Levy and Andrew, 2006), a tree-transformation utility built on TRegex. The scripts normalize passive verbs and past participial modifiers into their active forms. Each sentence is transformed before pattern extraction or matching.

TRegex Pattern Extraction

The algorithm for extracting TRegex patterns first preprocesses all training sentences with TSurgeon. Next, for each causal instance with both a cause and an effect, it uses the Dreyfus-Wagner algorithm (Dreyfus and Wagner, 1971) to find a minimum-weight subtree of the dependency graph that includes the connective, the cause head, and the effect head. The algorithm uses this to build the pattern: for each subtree edge $rel(A, B)$, the pattern requires that the test sentence include nodes related by a dependency labeled as rel. If A or B is a connective word, then the pattern also checks for its lemma and POS in the test sentence nodes. Patterns with more than six non-argument, non-connective nodes are discarded as parse errors or otherwise ungeneralizable flukes.

The graph for Dreyfus-Wagner includes a directed

[6]The actual TRegex pattern encoding these conditions is: /^because_[0-9]+$/=connective_0 <2 /^IN.*/ <1 mark >(/.*_[0-9]+/=cause <1 advcl >(/.*_[0-9]+/=effect)).

edge for each dependency in the sentence's parse tree. For each edge, a back-edge of equal weight is added, unless the back-edge already exists (which UD allows). This allows Dreyfus-Wagner to find a subtree even when it has to follow an arc in reverse to do so.

Most edges have unit weight. However, for nodes with multiple parents (which UD also allows), the algorithm would often choose the wrong dependency path, leading to poor generalization. Accordingly, on paths of the form $x \xrightarrow{\text{xcomp}} y \xrightarrow{\text{csubj} \mid \text{nsubj}} z$ (where xcomp indicates open clausal complements), edge costs are slightly decreased. This helps the algorithm prefer the xcomp path connecting x, y, and z, rather than a path such as $x \xrightarrow{\text{nsubj}} z \xleftarrow{\text{nsubj}} y$. Similarly, acl (adjectival clause) and expl (expletive) edges are slightly penalized.

Syntax-Based Argument Identification

Each syntactic pattern inherently encodes the positions in the tree of the cause and effect heads. Thus, matching these patterns is the first step of argument ID, in addition to (tentative) connective discovery.

The second step of argument ID is to expand the argument heads into complete spans. In general, most syntactic dependents of an argument's head are included in its span. There are two exceptions:

1. **Connective words.** Under the UD scheme, words that form part of the connective sometimes appear as dependents of the argument head. For example, in A *prevents* B *from* C, *from* appears as a dependent of C, but it is really part of the construction for the verb *prevent*. Following the annotation scheme, we therefore exclude such connective words (and any of their dependents) from the argument span.
2. **Words below the head of the other argument.** For example, in A *because* B, B will be a dependent of A (see Figure 1). Obviously, however, it should not be included in the cause span.

4.2 Causeway-L: Lexical Pattern-Based Tagging

Syntactic parsers often make mistakes, which becomes especially relevant for syntactic patterns that examine multiple dependencies. This is particularly problematic for the syntax-based pipeline, for which parse errors, either in training or at test time, can prevent patterns from matching. Additionally, the exact syntactic relations present in a given instance may be altered by the presence of other constructions. For example, if a verb appears as a complement of another verb, the path to the subject will have an additional dependency link.

We therefore implemented a second algorithm, Causeway-L, that performs connective discovery based on the sequence of word lemmas and parts of speech: instead of extracting and matching parse patterns, it extracts and matches regular expressions. It then uses a conditional random field to label the argument spans, using features from the parse in a more probabilistic way.

Connective Discovery with Regular Expression Patterns

At training time, we generate regular expressions that will match sequences of connective and argument lemmas. The regexes also make sure that there are tokens in the correct lexical positions for arguments. For instance, upon seeing an example like *A because B*, it would generate a pattern that matches any sequence of lemmas (the effect range), followed by the lemma *because* with POS tag IN (the connective), followed by any other sequence of lemmas (the cause range).[7] Each subpattern is given its own capturing group in the regular expression. At test time, each new sentence is turned into a string of lemmas with POS tags for matching. Matching lemmas can be recovered from the capturing groups.

Argument Identification with a CRF

Unlike syntactic patterns, regular expressions cannot pinpoint the cause and effect arguments; they can only give ranges within which those arguments must appear. Thus, the argument ID stage for this pipeline starts with much less information.

We treat the task of argument ID as a sequence labeling problem: given a particular regex-derived connective, the system must identify which tokens are in the cause, which are in the effect, and which are neither. We use a standard linear-chain CRF for this task, implemented with the CRFsuite library.[8]

[7] The actual regular expression that encodes this is: (^|)([\S]+)+?(because/IN) ([\S]+)+?.

[8] http://www.chokkan.org/software/

- The lemma of w_i
- The POS tag of w_i
- Whether $w_i \in C$
- The dependency parse path between w_i and the token in C that is closest in the parse tree
- The absolute lexical distance between w_i and the lexically closest token in C
- The signed lexical distance between w_i and the lexically closest token in C
- Whether w_i is in the parse tree (false for punctuation and several other non-argument token types)
- The regex pattern that matched C
- The cross-product of the regex pattern feature and the parse path feature
- The position of w_i relative to C
- Whether the lemma of w_i is alphanumeric

Table 4: Features used for CRF argument ID. For each possible connective C (a set of tokens), features are extracted from each word w_i in the sentence. All non-numeric features are binarized.

The features for argument ID are listed in Table 4.

4.3 Voting Classifier for Filtering

The pattern-matching stage overgenerates for two reasons. First, due to ambiguity of both words and constructions, not all instances of a given pattern are actually causal. *Since*, for example, has both causal and temporal senses, which are not distinguished either lexically or syntactically. Second, the patterns do not filter for important constructional elements like tense and modality (e.g., example 8 in Table 1 requires modality of necessity in the cause). Thus, pattern matching alone would yield high recall but low precision.

To account for this, the final stage of both pipelines is a filter that determines whether each possible connective instance, along with its arguments, is in fact being used in a causal construction.

This classification task is somewhat, but not entirely, heterogeneous. Some aspects are universal – there are regularities in what typically causes what –

Connective features:
- The label on the dependency from h to its parent
- The part of speech of h's parent
- The sequence of connective words*
- The sequence of connective lemmas*
- Each pattern that matched the connective*

Argument features:
- The POS tags of c and e
- The generalized POS tags of c and e (e.g., N for either NNP or NNS)
- The tenses[9] of c and e, both alone and conjoined
- The label on each child dependency of c and e
- For verbs, the set of child dependency labels
- The number of words between c and e
- The dependency path between c and e
- The length of the dependency path from c to e
- Each closed-class child lemma of e and of c
- The domination relationship between c and e (dominates, dominated by, or independent)
- The sets of closed-class child lemmas of e and c
- The conjoined NER tags of c and e
- Initial prepositions of the cause/effect spans, if any
- Each POS 1-skip-2-gram in the cause/effect spans
- Each lemma 1-skip-2-gram in the cause/effect spans that was seen at least 4 times in training
- Each WordNet hypernym of c and e†

Table 5: Features for the causal language candidate filter. c indicates the cause head, e the effect head, and h the connective head. All non-numeric features are binarized.
* Used only for per-connective classifiers.
† Used only for the global classifier.

while others are construction-dependent. To incorporate both kinds of information, a separate soft-voting classifier is created for each unique sequence of connective words. Each classifier averages the probability estimates of three less-reliable classifiers: a global logistic regression classifier, which is shared between all connectives; a per-connective logistic regression classifier; and a per-connective classifier that chooses the most frequent label for that connective. Our classifier thus differs slightly from typical voting classifiers: rather than the ensemble consisting of multiple algorithms trained on the same data, our ensemble includes one generalist and two specialists.

We use the scikit-learn 0.17.1 (Pedregosa et al., 2011) implementation of logistic regression with L1 regularization and balanced class weights. The logistic regression classifiers consider a variety of features

crfsuite/. We use the python-crfsuite wrapper (https://github.com/tpeng/python-crfsuite/).

[9]For this purpose, we represent the tense of a verb as the POS of the verb plus the string of auxiliaries attached to it. The tense of a non-verb is null. For copulas, both the POS of the copula and the POS of its object are included.

derived from the matched pattern, the connective words matching, the argument heads, and the parse tree (see Table 5). An instance is tagged as causal if the soft vote assigns it a probability above 0.45. This cutoff, close to the prior of 0.5, was tuned for F_1 on a different random split of the data than was used in the experiments below. In our experiments, the cutoff made little difference to scores.

5 Experiments

5.1 Baselines

Our task differs significantly from existing tasks such as frame-semantic parsing, both in the forms of allowable triggers and in the semantic relationships targeted. Our results are therefore not directly comparable to a frame-semantic parsing baseline. Instead, we compare our end-to-end results against an argument-aware most-frequent-sense (MFS) baseline.

At training time, the baseline first extracts the set of sequences of connective lemmas – i.e., $\{\langle \text{prevent, from} \rangle, \langle \text{because, of} \rangle, \ldots\}$. It then builds a table t with one entry for each combination of connective and argument parse path, recording how many more times it has been causal than non-causal.

For example, consider the sentence *The flu prevented me from attending*. After finding that *prevent* and *from* are present in the correct order, the baseline considers every pair (c, e) of non-connective words within a parse radius of two links from the connective. Each (c, e) is considered a possible cause/effect head pair for *prevent from*. For each pair, it finds the shortest path of dependency links from either *prevent* or *from* to c and e (call these paths d_c and d_e). If *prevent from* is annotated as causal, with cause head c and effect head e, the system increments the count $t\{\text{prevent from}, d_c, d_e\}$; otherwise, it decrements it.

The test algorithm finds the same set of possible (connective, cause head, effect head) tuples in the test sentence. For each tuple, if the corresponding entry in t is greater than 0, the system tags a causal language instance. Argument heads are expanded into spans using the algorithm from Causeway-S.

In addition to an end-to-end baseline, we wished to test how helpful our three-way voting classifier is. For each pipeline, then, we also compare that classifier against a most-frequent-sense baseline that chooses the most frequent label (causal or not-causal) for each connective, with connectives differentiated by their lemma sequences. The baseline classifier has no access to any information about the arguments.

5.2 Experiment 1: Pipeline Comparison

In this experiment, we measured the performance of Causeway-S, Causeway-L, and the baseline on the tasks of connective discovery and argument ID. We also tried taking the union of each system's outputs with the baseline's. Because of the small size of BECAUSE, we report averaged metrics from 20-fold cross-validation, with fold size measured by sentence count. All pipelines were run on the same folds.

Evaluation Metrics

For connective discovery, we report precision, recall, and F_1 for connectives, requiring connectives to match exactly. In counting true positives and false negatives, the only gold-standard instances counted are those with both a cause and an effect, in keeping with the task definition.

For argument ID, we split out metrics by causes and effects. For each, we report:

- percent agreement on exact spans
- percent agreement on heads
- the average Jaccard index (Jaccard, 1912) for gold-standard vs. predicted spans, defined as $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$, where A and B are the sets of tokens in the two spans. This metric reflects how well the argument spans overlap when they do not match exactly.

All argument metrics are reported only for correctly predicted connectives, as there is no way to automatically evaluate argument spans for false positives. Note that as a result, argument ID scores are not directly comparable between pipelines – the scores represent how well argument ID works given the previous stage, rather than in an absolute sense.

We use the same metrics for Experiments 2 and 3.

5.3 Experiment 2: Ablation Studies

Our second experiment explores the impact of various design choices by eliminating individual design elements. We report results from using the global and connective-specific classifiers on their own, without the soft-voting ensemble. (The MFS classifier is

tested alone as part of Experiment 1.) We also report results from the ensemble classifier without using any features that primarily reflect world knowledge: NER tags, WordNet hypernyms, and lemma skip-grams.

5.4 Experiment 3: Effects of Parse Errors

In our third experiment, we examined the effects of parse errors on our pipelines' performance. We compared each pipeline's performance with and without gold-standard parses, using only the Penn Treebank portion of BECAUSE. For gold-standard runs, we used the Stanford dependency converter (De Marneffe et al., 2006) to convert the gold parses into dependency format. We report averaged results from 20-fold cross-validation on this subcorpus.

6 Experimental Results and Analysis

6.1 Experiment 1 Results

Results from Experiment 1 are shown in Table 6.

Our most important conclusion from these results is that a classifier can indeed learn to recognize many of the subtleties that distinguish causal constructions from their similar non-causal counterparts. Even our end-to-end baseline offers moderate performance, but Causeway-L outperforms it at connective discovery by over 14 F_1 points, and Causeway-S outperforms it by 18 points.

The design of the filter is a significant contributor here. The MFS classifier alone substantially underperforms the voting classifier, particularly for the syntactic pipeline. The small connectives filter makes up some of the difference, but the full pipeline still beats the MFS filter by 4.4 points for the lexical system and 6.7 points for the syntax-based system.

When our pipelines are combined with the end-to-end baseline, the results are better still, beating the baseline alone by 21.4 points. This supports our hypothesis that causal construction recognition rests on a combination of both shallow and deep information.

As expected, both pipelines show high recall but low precision for the connective discovery stage. (Much of the remaining gap in recall came simply from the long tail of constructions – about half of connective types never saw a pattern match.) The filter does balance out precision and recall for a better F_1. However, as the filter's steep drop in recall suggests, more work is needed to upweight positive instances. Examining the classifier scores reveals that the filter is doing a good job of assigning low probability to negative instances: the vast majority of false pattern matches are clustered below a probability of 0.5, whereas the positives are peppered more evenly over the probability spectrum. Unfortunately, the positives' probabilities are not clustered on the high end, as they should be.

Significant leverage could be gained just from improving classification for the connective *to*. For both pipelines, this one connective accounted for 20–25% of end-to-end false positives and false negatives, and nearly half of all misclassifications by the filter. Many of the remaining errors (about 40%) came from just a few simple but highly ambiguous/polysemous connectives, including *if*, *for*, and *so*. For complex constructions (MWEs or more complex syntactic structures), Causeway-L achieved 42% F_1 and Causeway-S achieved 48%. Overall, then, it seems that the classifier is doing well even at the cases that would challenge typical semantic parsing systems, but it needs some features that will allow it to upweight positive instances of a few challenging words.

For argument ID, both techniques do reasonably well at recovering exact argument spans, and the H_C and H_E columns show that even when the exact spans do not match, the key content words are mostly correct. The low Jaccard indices, meanwhile, indicate that there is plenty of room for improvement in finding not just heads, but full spans.

Interestingly, effects seem to be harder to recover than causes. The likely culprit is the difference in lengths between the two types of arguments. The distribution of cause lengths is skewed toward low numbers, with a peak at 2 and a median of 5, while effects have a smoother peak at 5 with a median of 7 (Figure 2). The difference makes it harder for the system to guess full effect spans, and even for heads there are more plausible options. The length disparity, in turn, is probably due to the fact that causes are likely to be subjects (19%) or nominal modifiers (13%), which skew short, whereas most effects are primary clauses (24%), complements (30%), or direct objects (12%), which are often more complex.

6.2 Experiment 2 Results

Results for Experiment 2 are shown in Table 7.

The results using single classifiers, rather than an

Pipeline	Connectives			Causes			Effects		
	P	R	F_1	S_C	H_C	J_C	S_E	H_E	J_E
Causeway-S w/o classifier	7.3	**71.9**	13.2	65.0	84.3	39.3	30.4	63.0	30.7
Causeway-S w/ MFS	40.1	37.9	38.6	71.0	87.6	42.0	34.3	64.4	31.9
Causeway-S w/ MFS + SC filter	**60.9**	36.2	45.1	75.1	92.3	42.9	40.7	75.2	35.8
Causeway-S w/ classifier	51.9	47.6	49.4	68.7	86.9	39.9	38.0	72.5	34.1
Causeway-S w/ classifier + SC filter	57.7	47.4	**51.8**	67.1	84.4	39.0	37.7	70.7	33.4
Causeway-L w/o classifier	8.1	**91.1**	14.8	56.8	67.6	33.1	39.5	59.4	30.9
Causeway-L w/ MFS	61.2	34.8	44.1	73.9	84.7	42.3	51.2	74.3	37.6
Causeway-L w/ MFS + SC filter	**61.3**	33.9	43.5	74.5	85.0	42.5	50.8	74.0	37.4
Causeway-L w/ classifier	59.0	40.6	**47.9**	74.5	85.9	42.7	53.3	76.1	38.2
Causeway-L w/ classifier + SC filter	60.4	39.9	**47.9**	74.3	85.8	42.6	53.3	76.4	38.2
Baseline	**88.4**	21.4	33.8	74.1	94.7	43.7	48.4	83.3	38.4
Baseline + Causeway-S	59.6	**51.9**	**55.2**	67.7	85.8	39.5	39.5	73.1	34.2
Baseline + Causeway-L	62.3	45.2	52.3	73.6	88.9	42.8	53.9	78.6	38.7

Table 6: Results for Experiment 1. S_C and S_E indicate exact span match for causes and effects, respectively; H_C and H_E indicate percentage accuracy for cause and effect heads; and J_C and J_E indicate cause and effect Jaccard indices. "SC filter" indicates the filter for smaller overlapping connectives. For the combinations of the baseline with Causeway, the union of the baseline's and Causeway's outputs was passed to the SC filter.

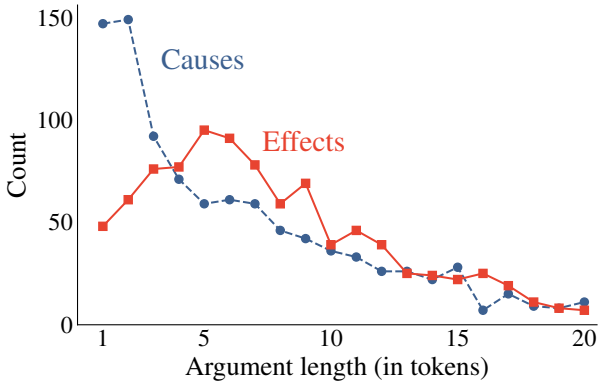

Figure 2: Distributions of cause and effect span lengths in the BECAUSE corpus (for instances with both arguments).

ensemble, uphold our design of combining multiple information sources. Even the best non-ensemble filter, the per-connective filter in Causeway-S, underperforms its ensemble counterpart by 3.1 points.

Likewise, the results from removing world-knowledge features confirm that this knowledge significantly assists the classifier beyond what surface-level features alone can provide. World knowledge features add 2.4 points for Causeway-L and 2.8 points for Causeway-S.

6.3 Experiment 3

Results from Experiment 3 are shown in Table 8.

As expected, Causeway-S improved significantly with gold-standard parses, whereas Causeway-L gets only a tiny boost. Surprisingly, Causeway-S did not improve from better TRegex matching of connectives per se. In fact, all scores for connective matching from the first stage were worse with gold-standard parses. Instead, the improvement appears to come from argument identification: better parses made it easier to identify argument heads, which in turn made the many features based on those heads more reliable. This is supported by the high argument head accuracy with gold parses. Further, when we ran the baseline on the PTB subcorpus with and without gold parses, we saw a similar improvement.

Thus, although the limited data and the classifier's failure to upweight positives are still the primary handicaps, better parses would be somewhat helpful for at least the syntax-based approach.

7 Related Work

Our work is of course based on CxG, which has inspired a number of NLP efforts. On the language-resource side, the FrameNet group, noting the many aspects of meaning that are not fully captured in an analysis of lexical triggers, has begun an extensive project to document and annotate grammatical con-

Pipeline	Classifiers Ablated	Connectives P	Connectives R	Connectives F_1	Causes S_C	Causes H_C	Causes J_C	Effects S_E	Effects H_E	Effects J_E
Causeway-S	–	**57.7**	47.4	**51.8**	67.1	84.4	39.0	37.7	70.7	33.4
Causeway-S	Both per-connective	40.8	50.4	44.9	65.6	82.8	38.1	36.8	68.8	32.6
Causeway-S	Global/most-freq.	47.1	**51.2**	48.7	66.0	82.3	38.3	36.3	69.1	33.0
Causeway-S	Knowledge features	49.5	49.0	49.0	68.9	85.1	39.8	38.3	71.7	33.4
Causeway-L	–	**60.4**	39.9	**47.9**	74.3	85.8	42.6	53.3	76.4	38.2
Causeway-L	Both per-connective	43.6	**40.0**	41.5	74.5	88.5	42.4	53.9	76.4	38.4
Causeway-L	Global/most-freq.	46.3	38.9	42.1	75.6	86.8	43.1	51.3	74.6	37.6
Causeway-L	Knowledge features	55.5	38.9	45.5	74.8	87.3	42.9	52.1	75.5	37.7

Table 7: Results for Experiment 2. Unablated full-pipeline results from Table 6 are included for comparison.

(a) With automatically parsed data

Pipeline	Connectives P	Connectives R	Connectives F_1	Causes S_C	Causes H_C	Causes J_C	Effects S_E	Effects H_E	Effects J_E
Causeway-S w/o classifier	14.9	**73.3**	24.7	63.6	90.9	40.3	18.1	72.7	25.3
Causeway-S w/ classifier + SC filter	**54.7**	40.2	**45.7**	78.7	98.4	44.6	46.0	78.4	36.7
Causeway-L w/o classifier	9.3	**84.6**	16.7	59.4	68.5	33.1	43.2	62.1	31.8
Causeway-L w/ classifier + SC filter	**52.4**	37.2	**43.2**	72.9	84.5	40.0	52.3	73.4	35.7

(b) With gold-standard parses

Pipeline	Connectives P	Connectives R	Connectives F_1	Causes S_C	Causes H_C	Causes J_C	Effects S_E	Effects H_E	Effects J_E
Causeway-S w/o classifier	10.2	**70.6**	17.7	79.4	98.1	45.7	52.8	90.2	41.3
Causeway-S w/ classifier + SC filter	**62.7**	51.6	**56.0**	80.2	96.4	45.6	59.0	92.7	43.4
Causeway-L w/o classifier	9.1	**84.1**	16.4	57.8	68.2	33.3	53.0	68.0	34.4
Causeway-L w/ classifier + SC filter	**56.4**	37.9	**44.3**	77.0	85.3	41.8	67.2	83.4	40.4

Table 8: Results for Experiment 3.

structions in English (Fillmore et al., 2012). Similar efforts are underway for VerbNet (Bonial et al., 2011) and PropBank (Bonial et al., 2014). On the NLP-tools side, some work has been done on parsing text directly into constructions, particularly through the formalisms of Fluid Construction Grammar (Steels, 2012) and Embodied Construction Grammar (Bergen and Chang, 2005), which take a "constructions all the way down" approach. Some HPSG parsers and formalisms, particularly those based on the English Resource Grammar (Copestake and Flickinger, 2000; Flickinger, 2011) or Sign-Based Construction Grammar (Boas and Sag, 2012), also take constructions into account. Thus far, however, only a few attempts (e.g., Hwang and Palmer, 2015) have been made to integrate constructions with robust, broad-coverage NLP tools/representations.

Other aspects of our work are more closely related to previous NLP research. Our task is similar to frame-semantic parsing (Baker et al., 2007), the task of automatically producing FrameNet annotations. Lexical triggers of a frame correspond roughly to our causal connectives, and both tasks require identifying argument spans for each trigger. The tasks differ in that FrameNet covers a much wider range of semantics, with more frame-specific argument types, but its triggers are limited to lexical units, whereas we permit arbitrary constructions. Our multi-stage approach is also loosely inspired by SEMAFOR and subsequent FrameNet parsers (Das et al., 2014; Roth

and Lapata, 2015; Täckström et al., 2015).

Several representational schemes have incorporated elements of causal language. PDTB includes reason and result relations; FrameNet frames often include Purpose and Explanation roles; preposition schemes (e.g., Schneider et al., 2015, 2016) include some purpose- and explanation-related senses; and VerbNet and PropBank include verbs of causation. As described in §1, however, none of these covers the full range of linguistic realizations of causality.

The ASFALDA French FrameNet project recently proposed a reorganized frame hierarchy for causality, along with more complete coverage of French causal lexical units (Vieu et al., 2016). Some constructions would still be too complex to represent, but under their framework, many of our insights could likely be merged into mainline English FrameNet.

Other projects have attempted to address causality more specifically. For example, a small corpus of event pairs conjoined with *and* has been annotated as causal or not causal (Bethard and Martin, 2008), and a classifier was built for such pairs (Bethard et al., 2008). The CaTeRS annotation scheme (Mostafazadeh et al., 2016), based on TimeML, also includes causal relations, but from a commonsense reasoning standpoint rather than a linguistic one. A broader-coverage linguistic approach was taken by Mirza and Tonelli (2014). They enriched TimeML to include causal links and their lexical triggers, and built an SVM-based system for predicting them. Their work differs from ours in that it requires arguments to be TimeML events; it requires connectives to be contiguous spans; and their classifier relies on gold-standard TimeML annotations.

More recently, Hidey and McKeown (2016) automatically constructed a large dataset with PDTB-style AltLex annotations for causality. Using this corpus, they achieved high accuracy in finding causality indicators. This was a somewhat easier task than ours, given their much larger dataset and that they limited their causal triggers to contiguous phrases. Their dataset and methods for constructing it, however, could likely be adapted to improve our systems.

Our pattern-matching techniques are based on earlier work on LEXICO-SYNTACTIC PATTERNS. These patterns, similarly represented as fragments of dependency parse trees with slots, have proven useful for hypernym discovery (Hearst, 1992; Snow et al., 2005). They have also been used both for the more limited task of detecting causal verbs (Girju, 2003) and for detecting causation relations that are not exclusively verbal (Ittoo and Bouma, 2011).

Our work extends this earlier research in several ways. We propose several methods (CRF-based argument ID and statistical classifiers) for overcoming the ambiguity inherent in such patterns. We also take care to ground our notion of causality in a principled annotation scheme for causal **language**. This avoids the difficulties of agreeing on what counts as real-world causation (see Grivaz, 2010).

8 Conclusion and Future Work

With this work, we have demonstrated the viability of two approaches to tagging causal constructions. We hope that the constructional perspective will prove applicable to other domains, as well. Our code and corpus are available at `https://github.com/duncanka/causeway` and `https://github.com/duncanka/BECauSE`, respectively.

In the immediate future, we plan to explore more sophisticated, flexible algorithms for tagging causal constructions that rely less on fixed patterns. Two promising directions for flexible matching are tree kernels and parse forests (Tomita, 1985). We are also pursuing a neural, transition-based tagging model.

In parallel, we are working to extend our approaches to cases where causality is expressed using temporal language or other overlapping relations. We are developing a further expanded corpus that will include annotations for such cases, and we expect to extend our algorithms to these new annotations.

In the longer run, we plan to demonstrate the usefulness of our predicted causal language annotations for an application-oriented semantic task such as question-answering.

Acknowledgments

We thank Jeremy Doornbos, Donna Gates, Nora Kazour, Chu-Cheng Lin, Michael Mordowanec, and Spencer Onuffer for all their help with refining the annotation scheme and doing the annotation work. We are also grateful to Nathan Schneider for his invaluable suggestions, and to the anonymous reviewers and TACL editors for their useful feedback.

References

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval'07 task 19: frame semantic structure extraction. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 99–104. Association for Computational Linguistics, Prague, Czech Republic.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th International Conference on Computational Linguistics*, volume 1, pages 86–90. Association for Computational Linguistics, Montreal, Canada.

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*, volume 2, pages 267–292. CRC Press, Boca Raton, FL.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510. Association for Computational Linguistics, Doha, Qatar.

Benjamin Bergen and Nancy Chang. 2005. Embodied construction grammar in simulation-based language understanding. *Construction grammars: Cognitive grounding and theoretical extensions*, 3:147–190.

Steven Bethard, William J Corvey, Sara Klingenstein, and James H. Martin. 2008. Building a corpus of temporal-causal structure. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, pages 908–915. European Languages Resources Association, Marrakech, Morocco.

Steven Bethard and James H. Martin. 2008. Learning semantic links from a corpus of parallel temporal and causal relations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies (ACL-08 HLT): Short Papers*, pages 177–180. Association for Computational Linguistics, Columbus, Ohio.

Hans Christian Boas and Ivan A. Sag, editors. 2012. *Sign-based construction grammar*. CSLI Publications, Stanford, CA.

Claire Bonial, Julia Bonn, Kathryn Conger, Jena D. Hwang, and Martha Palmer. 2014. PropBank: Semantics of new predicate types. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 3013–3019. European Languages Resources Association, Reykjavik, Iceland.

Claire Bonial, Susan Windisch Brown, Jena D. Hwang, Christopher Parisien, Martha Palmer, and Suzanne Stevenson. 2011. Incorporating coercive constructions into a verb lexicon. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 72–80. Association for Computational Linguistics, Portland, Oregon.

Juliette Conrath, Stergos Afantenos, Nicholas Asher, and Philippe Muller. 2014. Unsupervised extraction of semantic relations using discourse cues. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 2184–2194. Dublin City University and Association for Computational Linguistics, Dublin, Ireland.

Ann A Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*, pages 591–600. European Language Resources Association, Athens, Greece.

Dipanjan Das, Desai Chen, André F.T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D. Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, volume 6, pages 449–454. European Languages Resources Association, Genoa, Italy.

Stuart Dreyfus and Robert Wagner. 1971. The Steiner problem in graphs. *Networks*, 1(3):195–207.

Jesse Dunietz, Lori Levin, and Jaime Carbonell. 2015. Annotating causal language using corpus lexicography of constructions. In *Proceedings of The 9th Linguistic Annotation Workshop (LAW IX)*, pages 188–196. Association for Computational Linguistics, Denver, CO.

Charles J. Fillmore. 2012. Encounters with language. *Computational Linguistics*, 38(4):701–718.

Charles J. Fillmore, Paul Kay, and Mary Catherine O'Connor. 1988. Regularity and idiomaticity in grammatical constructions: The case of *let alone*. *Language*, 64(3):501–538.

Charles J. Fillmore, Russell Lee-Goldman, and Russell Rhodes. 2012. *Sign-based construction grammar*, chapter The FrameNet constructicon, pages 309–372. In Boas and Sag (2012).

Dan Flickinger. 2011. Accuracy vs. robustness in grammar engineering. In Emily M. Bender and Jennifer E. Arnold, editors, *Language from a cognitive perspective: Grammar, usage, and processing*, volume 201 of *CSLI Lecture Notes*, pages 31–50. CSLI Publications.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering*, volume 12, pages 76–83. Association for Computational Linguistics, Sapporo, Japan.

Adele Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. Chicago University Press, Chicago, IL.

Cécile Grivaz. 2010. Human judgements on causation in French texts. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 2626–2631. European Languages Resources Association, Valletta, Malta.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics*, volume 2, pages 539–545. Association for Computational Linguistics, Nantes, France.

Christopher Hidey and Kathleen McKeown. 2016. Identifying causal relations using parallel Wikipedia articles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1424–1433. Association for Computational Linguistics, Berlin, Germany.

Jena D. Hwang and Martha Palmer. 2015. Identification of caused motion constructions. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (* SEM 2015)*, pages 51–60. Association for Computational Linguistics, Denver, CO.

Ashwin Ittoo and Gosse Bouma. 2011. Extracting explicit and implicit causal relations from sparse, domain-specific texts. In *Proceedings of the 16th International Conference on Natural Language Processing and Information Systems (NLDB '11)*, pages 52–63. Springer-Verlag, Alicante, Spain.

Paul Jaccard. 1912. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 423–430. Association for Computational Linguistics, Sapporo, Japan.

Lori Levin, Teruko Mitamura, Davida Fromm, Brian MacWhinney, Jaime Carbonell, Weston Feely, Robert Frederking, Anatole Gershman, and Carlos Ramirez. 2014. Resources for the detection of conventionalized metaphors in four languages. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 498–501. European Language Resources Association, Reykjavik, Iceland.

Roger Levy and Galen Andrew. 2006. TRegex and TSurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 2231–2234. European Language Resources Association, Genoa, Italy.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating

predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119. Association for Computational Linguistics, Plainsboro, NJ.

Paramita Mirza and Sara Tonelli. 2014. An analysis of causality between events and its relation to temporal information. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 2097–2106. Dublin City University and Association for Computational Linguistics, Dublin, Ireland.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the 4th Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 51–61. Association for Computational Linguistics, San Diego, CA.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association, Portoro, Slovenia.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*, pages 2961–2968. European Language Resources Association, Marrakech, Morocco.

Michael Roth and Mirella Lapata. 2015. Context-aware frame-semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:449–460.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium*.

Jonathan Schaffer. 2014. The metaphysics of causation. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2014 edition. `http://plato.stanford.edu/archives/sum2014/entries/causation-metaphysics/`.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Meredith Green, Abhijit Suresh, Kathryn Conger, Tim O'Gorman, and Martha Palmer. 2016. A corpus of preposition supersenses. In *Proceedings of the 10th Linguistic Annotation Workshop (LAW X)*, pages 99–109. Association for Computational Linguistics, Berlin, Germany.

Nathan Schneider, Vivek Srikumar, Jena D. Hwang, and Martha Palmer. 2015. A hierarchy with, of, and for preposition supersenses. In *Proceedings of The 9th Linguistic Annotation Workshop (LAW IX)*, pages 112–123. Association for Computational Linguistics, Denver, CO.

Karin K. Schuler. 2005. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA. AAI3179808.

Noah A. Smith, Claire Cardie, Anne Washington, and John Wilkerson. 2014. Overview of the 2014 NLP unshared task in poliinformatics. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 5–7. Association for Computational Linguistics, Baltimore, MD.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17 (NIPS 2004)*, pages 1297–1304. MIT Press, Vancouver, Canada.

Luc Steels, editor. 2012. *Computational Issues in Fluid Construction Grammar*. Lecture Notes in

Computer Science. Springer Verlag, Berlin, Germany.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Masaru Tomita. 1985. An efficient context-free parsing algorithm for natural languages. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 756–764. Morgan Kaufmann Publishers Inc., Los Angeles, CA.

Laure Vieu, Philippe Muller, Marie Candito, and Marianne Djemaa. 2016. A general framework for the annotation of causality based on FrameNet. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3807–3813. European Language Resources Association, Portoro, Slovenia.

Phillip Wolff, Bianca Klettke, Tatyana Ventura, and Grace Song. 2005. Expressing causation in English and other languages. In Woo-kyoung Ahn, Robert L. Goldstone, Bradley C. Love, Arthur B. Markman, and Phillip Wolff, editors, *Categorization inside and outside the laboratory: Essays in honor of Douglas L. Medin*, pages 29–48. American Psychological Association, Washington, DC.