Populating a Knowledge Base with Information about Events

Sean Monahan, Michael Mohler, Marc Tomlinson, Amy Book,
Maxim Gorelkin, Kevin Crosby, Mary Brunson
Language Computer Corporation
smonahan@languagecomputer.com

Abstract

This paper describes a system for populating a knowledge base (KB) with events that are identified in text. We first describe how mentions of an event can be discovered in a document by utilizing lexical priors with contextual sense disambiguation, and then describe a machine learning model which identifies the value of the realis attribute for each mention. Finally, a multi-constraint architecture is used for linking the event mentions within a document into groups of coreferential events or *event hoppers*. The grouped event mentions begin to form a rich event structure within a document that can then be incorporated into a KB.

1 Introduction

In this paper, we present our work aimed at populating a knowledge base (KB) with rich information about events. The ability to acquire information about an event from text, whether it be an election, battle, product launch, or court case, is profoundly important to understanding the state of the world. The goal of the Text Analysis Conference (TAC) Knowledge Base Population (KBP) evaluation is to research novel approaches to populating knowledge bases, especially with information about entities and events. In the following sections, we describe a system for the newly introduced 2015 TAC KBP *Event Detection and Coreference* tasks.

The three components of the system are responsible for (1) detecting event mentions for a set of given event types, (2) associating event mentions with realis attributes, and (3) linking coreferential event mentions into clusters called *event hoppers*. The example below reflects the execution of the three components over a text, which are detailed in the paragraphs that follow.

"Yesterday, Google $announced^1$ (B,ACTUAL) the upcoming $acquisition^2$ (M,OTHER) of one of their main rivals. The $merger^2$ (M,OTHER) will net^3(T,OTHER) the company's CEO over $100 million. The $announcement^1$ (B,ACTUAL) comes only days after a failed $acquisition^4$ (T,OTHER)."

The first component, which performs event detection, involves the identification and classification of event mentions. Spans of text that indicate a particular event (often verbal or nominal predicates) are called *triggers*. In the example, three types of events are associated with the italicized triggers: Contact.Broadcast (B), Business.Merge-Org (M), and Transaction.Transfer-Money (T). Determining which type of event is implied by a trigger is challenging because of the many ways to reference a particular event in text, as well as the ambiguity that can result when contextual cues within the document are not considered.

The second component identifies the value of the event's *realis* attribute. The realis attribute (henceforth 'realis') of an event refers to whether the mentioned event was a specific event that occurred (ACTUAL), one that did not occur (OTHER), or a general class of occurrences (GENERIC).

Finally, event mentions which refer to the same event must be grouped together into "hoppers". Most events that are interesting to real-world applications cannot typically be described by a single utterance or sentence. Instead, the event's ar-

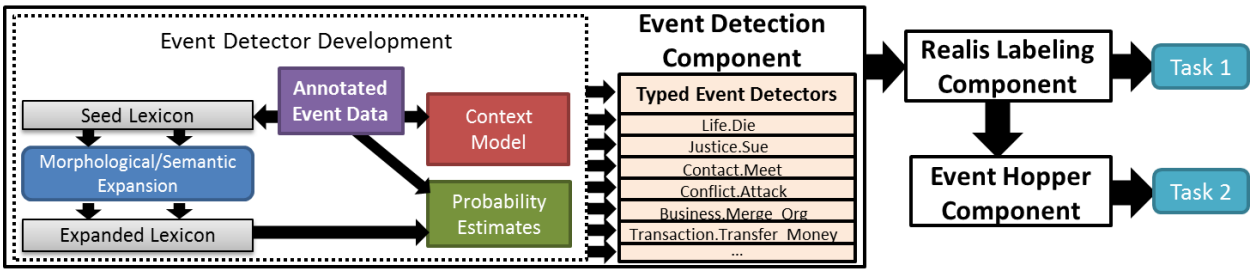

Figure 1: The full pipeline of our 2015 KBP Event Trigger Detection and Coreference submission.

guments (Agent, Time, Place, etc.), attributes (e.g. realis), relationships to sub-/super-events within an event structure, and relationships to events outside the structure are revealed across multiple mentions. Hoppers execute the important function of gathering all of this information about the event across the entire document. In the example paragraph above, the event hopper is indicated by the numerical superscript on the event mention. Note that the same event can be referenced using different triggers (e.g. hopper 2), and similar triggers can refer to different events (e.g. *acquisition*).

The TAC KBP 2015 Event Detection tasks evaluate systems' ability to perform the functions described above. This track consists of three separate tasks: Task 1 - *Event Detection*, Task 2 - *Event Detection and Coreference*, and Task 3 - *Event Coreference*. As the names indicate, the first task involves event and realis detection alone, the second task combines event detection and realis association with placement into hoppers, and the third task requires systems to group gold event triggers that are provided by the LDC into hoppers. A diagram of the complete system we used for the various components of these tasks is shown in Figure 1.

2 Related Work

The KBP Event Detection and Coreference tasks build on work being done in the areas of event extraction, event attribute labeling, and event coreference. Event extraction typically starts with the concept of an event trigger, which is defined in LDC (2014a) as "the smallest extent of text (usually a word or short phrase) that expresses the occurrence of an event". An alternative type of event mention is Mitamura et al.'s (2015) concept of "event nugget", which represents the *maximal* extent of a textual event indicator. Accordingly, event nuggets encourage the selection of the full semantic breadth of an event mention in text, as opposed to triggers, which favor brevity over complete semantic expression. While recent event detection annotation schemas have explored the use of both of these kinds of event mentions, the spans extracted in this year's tasks utilize event triggers.

Central to the notion of an event trigger is what it means for an event to actually be indicated in text, an area covered in Monahan and Brunson (2014), which explored the concept of *eventiveness* and its role in differing definitions of event extraction. Eventiveness describes the degree to which a span of text can be considered to indicate an event. Verbs such as "protested", "jailed", and "died" are typically strongly eventive, as are many nouns, such as "wedding" and "attack". Other word classes, such as adjectives ("*married* man"), personal descriptors ("protester"), prepositional phrases ("behind bars"), and object descriptors ("*bomber* plane") tend to be only weakly eventive.

There is no fixed rule for determining the boundary between "event" and "non-event"; indeed, this threshold can vary greatly among use-cases and annotators. Thus, for the KBP tasks, it is important to lean heavily on annotated data to determine where this boundary lies. As such, of the three general approaches to event extraction that are described by Hogenboom et al. (2011) — data-driven, expert knowledge-driven, and hybrid — the 2015 Event Detection task requires a data-driven approach, even though hybrid approaches have proven successful for previous tasks (Monahan et al., 2014).

Labeling event attributes has received significant interest in the last several years. A wide variety of

different event attributes have been examined, including polarity (Morante and Blanco, 2012), genericity (Mathew and Katz, 2009; Reiter and Frank, 2010), factuality (Saurí and Pustejovsky, 2009; de Marneffe et al., 2012), and author perspectives (Bracewell et al., 2014). Although the KBP realis status is more general than these attributes, it can be derived from combinations of attributes, as in Monahan et al. (2014).

Associating events into hoppers is similar to the procedure of event coreference. Early work in event coreference, such as that of Bejan and Harabagiu (2010), utilized an unsupervised approach with rich features. More recent work, such as that of Liu et al. (2014), has made use of recent annotation work to develop a supervised system for event coreference that propagates information between events and their arguments.

3 Data Sources

In order to build our components for event detection, realis labelling, and event coreference, we leveraged a variety of different data sources. The LDC provided several resources, including E73 (2015c), E121 (2014b), and E29 (2015b). For training our event detection and realis systems, we split *allEN* into *trainEN*, which consists of 60% of each of the three, *devEN*, which was 20% of each, and *testEN* with the remainder. Since E121 did not possess hopper data, we only utilized E73 and E29 for training the event hopper system. We additionally derived data from other data sources such as TAC2014 (2015a) but only for background knowledge, as these data sources had not been specifically annotated for triggers, and hence were unsuitable for training data in the KBP trigger tasks.

4 Event Detection

Our event detection system is a pipeline consisting of event discovery, prior probability estimation, contextual probability, and event extraction. First, the event discovery module searches in text and finds the triggers that are possibly indicative of events. Then, we associate a prior probability with each lexical item, and subsequently employ a contextual system to determine whether the event type is being indicated correctly in the text. Finally, the event detection model combines these features to make the final determination to accept or reject each candidate trigger.

4.1 Event Trigger Lexicon

The event discovery module consists of an extensive lexicon of words and phrases that can indicate events. The purpose of this lexicon is to serve as a high-recall selector of text, which can then be further processed using other statistical modules.

To develop the lexicon, we first leveraged the event argument extractors that we developed for the KBP 2014 Event Argument Extraction (EAE) task (NIST, 2014). Each of these extractors had a lexicon of manually curated terms which were indicative of these event types. We developed additional extractors for event arguments for the ten new event types introduced in 2015. We then extended the lexicon with the annotated triggers for the 2014 EAE task, which were made available as part of LDC (2015a). Specifically, we added all of the words in any WordNet synset that contained two or more triggers for a given event type in the 2014 training data. Finally, for the evaluation, we also added the triggers from the *trainEN* portion of our data set.

During the event detection process, we attempt to match candidate triggers against our lexicon using the exact form found in the aforementioned lexicon sources, the porter-stemmed form, or any derived form[1]. For example, the lexical item "injure" can be matched to all possible inflections of the verb "to injure" as well as to the nominal form "injury". In total, the lexicon had approximately 31,000 items (including morphological variants).

4.2 Prior Probabilities

For each lexicon item, we estimate the prior probability that the item indicates a given event type E. We consider a variety of different estimates, based on the exact string S ("attacked") $P(E|S)$, exact string with part-of-speech ("attack#v" or "attack#n") $P(E|S, pos)$, lemma $P(E|lemma)$, and lemma with part-of-speech $P(E|lemma, pos)$.

To conduct this estimation, we utilize the human-annotated event triggers in *trainEN*. For example, if

[1] We used both the derivational relations in WordNet and resources from the NOMLEX project (Macleod et al., 1998) to map between verbs and nouns.

the word "attack" occurs 38 times in E73, and 34 of those are event mentions , then the prior probability for the string "attack" is 89.5%. Overall, counts were found for 2,868 of the lexicon items.

4.3 Contextual Probability

Lexical items with a high probability of being annotated as signaling an event ("death", "sentenced") see little benefit from the use of additional context for disambiguation. However, lexical items with low probabilities ("went", "have"), require significant contextual clues to disambiguate. Sometimes, these contextual cues are expressed as arguments that can be detected with a semantic parser, but in many instances, the disambiguating words have no direct semantic or syntactic relation to the event trigger, or they are too distant to be reliably linked. Therefore, we have created a context model for each event type that detects contextual cues without relying on a semantic parser.

To do this, we used a context-level neural embedding approach (Řehůřek and Sojka, 2010; Le and Mikolov, 2014), which creates a vector representation for an entire sentence. This extends Mikolov et al. (2013) by jointly learning a vector representation for the sentence along with the representations of the words within that sentence. Sentences with similar vectors in this representation thus have similar meanings. For each event type, we created a kernel density estimator for the relative likelihood of the event's presence or absence based on a set of context vectors from the training data. We utilized a product of Gaussian kernels to estimate the relative likelihood of a new context vector, x, being a positive instance of the event across each dimension, i, based on the set of vectors for positive event instances, $y \in \mathbf{E}+$.

$$\hat{f}_h(x, \mathbf{E}+) = \frac{1}{n} \sum_{y \in \mathbf{E}+} \prod_{i \in dim} \frac{1}{\sqrt{2\pi} h_i} e^{-\frac{(x_i - y_i)^2}{2h_i^2}}$$

The optimal bandwidth, h_i, was selected independently for each dimension through cross-validation. In addition, we estimated the relative likelihood of the event being contraindicated using vectors where the event is explicitly absent, but valid triggers are present $y \in \mathbf{E}-$. The sets $\mathbf{E}-$ and $\mathbf{E}+$ were taken from the gold data and from system responses for TAC2014 that had been reannotated (5,118 events in total) as being true-positives or true-negatives.

4.4 Event Detection Model

We combine the trigger-event probability estimate with the context-event probability estimate using a logistic regression model. The model utilizes the following features: (1) $P(E|S)$, (2) $P(E|lemma)$, (3) $P(E|S, pos)$, and (4) $P(E|lemma, pos)$, as defined in section 4.2, as well as (5) the relative likelihood of the event type given the sentence context vector $\hat{f}_h(x, \mathbf{E}+)$; and (6) the relative likelihood of a non-event given the sentence context vector, $\hat{f}_h(x, \mathbf{E}-)$. The logistic regression model was trained on *trainEN*.

For around half of the event types, our experiments showed that the system utilizing only the lexical probability feature (word only) produced the highest performance, and for those types, we did not utilize the ML model. Also note that we individually selected which system to utilize for each event type based not on the F-measure for that particular event type, but rather on the effect on the overall micro F-measure across all event types.

4.5 Other Event Detection Techniques

In addition to utilizing the statistical lexicon and contextual systems described above, we also capitalized on LCC's system for identifying the arguments of the event mentions in the KBP 2014 EAE task. Our previous approach (Monahan et al., 2014) consisted of an expert semantic pattern approach and an ML system called CiceroCustom. The expert semantic pattern system lacked sufficient recall for the 2015 tasks, given that many of the event triggers occurred without specific semantic frames. However, its argument extraction capabilities were sufficient for extracting the mentions' arguments for realis classification and event clustering.

5 Realis Labeling

Once detected, event mentions in text offer a wealth of information to the end user. One such element is the certainty with which the event can be understood to have occurred. The process of determining and selecting one of three values — ACTUAL, GENERIC, and OTHER — to capture this information is known as *realis labeling*. An ACTUAL

event refers to a specific event that has occurred, a GENERIC event refers to a general class of events, and an OTHER event refers to an event that has not occurred.

One set of event attributes that our system utilizes to perform realis labeling is the four attributes that are described in the ACE (2005) specification: *genericity*, *polarity*, *tense*, and *modality*. Genericity indicates whether the event mention describes a specific occurrence (or bounded set of such occurrences) or whether it represents a general class of occurrences, polarity refers to whether the event's occurrence is positively stated or explicitly negated, tense refers to the time of an event's occurrence, and modality refers to the level of certainty surrounding its occurrence as presented by the speaker. The relationships between the ACE event attributes and KBP realis labels are displayed in Table 1 below.

KBP	**ACE**			
Realis	Genericity	Polarity	Tense	Modality
ACTUAL	SPECIFIC	POSITIVE	PAST / PRESENT	ASSERTED
OTHER		NEGATIVE	ANY	ANY
		ANY	FUTURE	
GENERIC	GENERIC		ANY	

Table 1: Comparison of ACE and KBP event attributes.

For polarity, tense, and modality, we directly utilize models trained on the ACE data as features for the realis system. For genericity, there are several cases where this simple mapping to ACE is incorrect in practice. Thus, we trained the genericity model on *trainEN*, with GENERIC event triggers serving as positive examples, and triggers that were ACTUAL or OTHER serving as negative examples.

Monahan et al. (2014) used models for the four ACE event attributes as well as 14 additional heuristics to detect lexico-syntactic indicators of different realis values. In this work, these models and heuristics are incorporated into a single machine learning model as part of a suite of linguistic features that capture a variety of realis indicators, as shown in Table 2.

Group	**Features**
Lexical attributes	Word sense
Grammatical features	Voice, tense-aspect-mood, POS, number, person
Syntactic relations	Dependency relation, clausal head
Sentential components	Arguments, adjunct PP, determiner, modifiers
Discourse functions	Headline/prose, interrogative/declarative

Table 2: Realis Feature Groups

Additionally, each feature is configured to operate over a pre-defined scope, which is roughly the span of text considered when calculating that feature. The scopes that correspond with each feature are shown in Table 3 below.

Scope	**Features**
Trigger Phrase	Words or parts-of-speech of the trigger
Argument	Words, POS, or semantic category of the argument
Clause	Entire clause containing the trigger and arguments
Parent/Ancestor	Verb which dominates the trigger, trigger's parent, and so on
Sentence	Entire sentence level

Table 3: Realis Feature Scopes

In addition to the features listed above, we also utilized a set of higher-level attributes (genericity, modality, polarity, and veridicality) that were largely derived from features in Mathew and Katz (2009) and Reiter and Frank (2010) and also drew in the functionality of underlying models that were already in place in our system. Aside from the aforementioned sources, much of the literature that deals with these higher-level attributes also explores similar types of lexical, semantic, and syntactic phenomena that are embodied in our lower-level features.

One caveat of using the features above is that many of them, such as modal adverbs, are specific to verbal triggers, and notably, only 53.7% percent of event triggers in E73 are verbs. Of the non-verbal triggers, 26.7% percent of these are nominal triggers with semantic frames (as specified in NOMLEX), and the remaining triggers are eventive words that bear other parts of speech. To make use of features specifically designed for verbal triggers for other-POS triggers, we utilize a dependency parse to find the parent verb of the event trigger and then use that verb's features as the pseudo-features of the non-verbal trigger.

Algorithm 1 for event realis label extraction encompasses the machine learning models we designed to extract the four ACE event attributes. Algorithm 2 contains a machine learning (logistic regression) model that we trained using the four ACE models, including the re-trained genericity model). Algorithm 3 consists of a Random Forest[2] approach, which proved to be the best strategy on *testEN* after we experimented with a variety of different ML models to use for combining the full set of rich fea-

[2]Cognitive Foundry: http://foundry.sandia.gov/

tures. The results of these algorithms are shown in Table 4. Note that we additionally experimented with utilizing separate models for verbal and nominal triggers, but did not achieve noticeable gains as a result.

Method	F-Measure
Algorithm 1	68.18
Algorithm 2	69.43
Algorithm 3	70.83

Table 4: Results of our realis labels on gold events.

6 Event Linking into Hoppers

After extracting typed event triggers and identifying their realis labels, we then cluster individual event mentions that are intuitively equivalent into loosely coreferential groups called event hoppers. This differs from traditional event coreference only insofar as it permits a looser definition of equivalence, allowing (among other things) differing event arguments when facts are being debated, realis labels that change over time, and varying levels of trigger and argument specificity (not to be confused with event structure granularity).

Beginning with a collection of event mentions from a single document, we construct these hoppers by independently determining the compatibility of each pair of event mentions in the document by using a multi-stage pipeline (see Figure 2) that considers event type, realis, trigger- and argument-level compatibility, and discourse characteristics. For event argument extraction, we make use of an in-house semantic parser along with LCC's Evention and CiceroCustom argument extraction technologies, as described in our 2014 KBP submission paper (Monahan et al., 2014).

Once these pairwise-compatibility ratings have been determined, we employ a greedy iterative clustering algorithm to produce high-quality event hoppers that serve to model an event at the document level. In particular, we define an N by N compatibility matrix, where N represents the number of event mentions in a document. Each element in this matrix is filled with either (1) a binary variable indicating that the pair must not be included in the same hopper, or (2) a non-negative score indicating the overall level of compatibility between the event mentions. Once the matrix is filled, we begin the iterative process of building hoppers. In the beginning, each mention belongs to its own singleton hopper. Then, at each subsequent step, we greedily select the two mentions across different hoppers that possesses the highest positive score and attempt to merge their corresponding hoppers. If merging the hoppers would result in any pair of incompatible mentions being included in the same hopper, the selected pair is itself marked for non-inclusion. Otherwise, the hoppers are merged. This process ends when all positively-scored pairs have been considered.

In the following subsections, we discuss each of the following modules of our event hopper component as shown in Figure 2: event type compatibility, realis compatibility, trigger compatibility, argument-level compatibility, and discourse processing. For each module, we will (1) show the impact of that module independent of the modules further downstream in the pipeline, and (2) compare against two baselines: (a) keeping all event mentions as singletons, and (b) merging all mentions with the same event type.

We report three types of metrics. First, we report the official evaluation criterion for the 2015 TAC KBP event, which ***KBP*** represents as the average of four well-known coreference metrics used: BCUBED, CEAFE, MUC, and BLANC, as defined by NIST (2015). In addition, we report ***PairP*** and ***PairR***, which are the precision and recall over event mention pairs. More specifically, assuming that (1) GH is the number of event mention pairs in the gold-standard hoppers, (2) SH is the number of pairs in the system-generated hoppers, and (3) JNT is the number of system hopper pairs that are also paired in the gold hoppers, then $PairP = \frac{JNT}{SH}$ and $PairR = \frac{JNT}{GH}$. We evaluate the hopper system on our test data in two modes: a "primary" mode in which the triggers, event types, and realis labels have been identified by our event trigger system (as with TAC KBP Task 2); and a "diagnostic" mode, which uses gold standard triggers, event types, and realis labels provided by the LDC (as with TAC KBP Task 3).

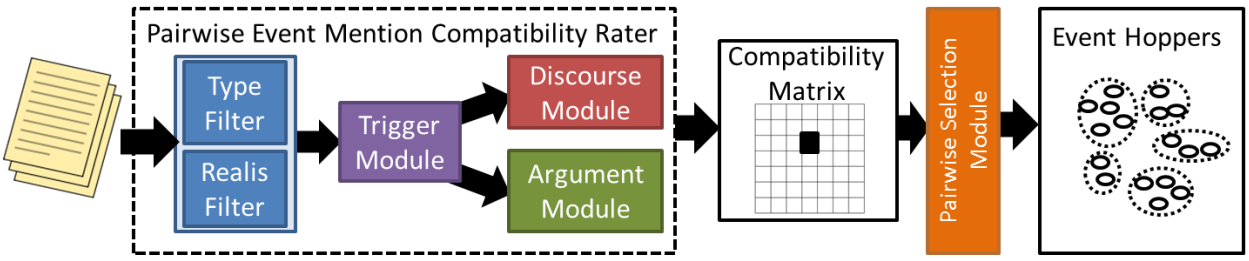

Figure 2: The five modules of our Event Hopper compatibility component along with the selection module.

6.1 Event Type Compatibility Module

Our first module makes a binary decision on whether a pair of event mentions can be in the same hopper by determining whether their event types are compatible. Only mentions with the same event type and subtype can be included in the same hopper, except for the underspecified *Contact.Contact* and *Transaction.Transaction* subtypes, which can be included with other *Contact* and *Transaction* mentions.

6.2 Realis Compatibility Module

Our second module determines whether the realis attributes are consistent between event mentions in the same hopper. It can often happen that two events with differing realis are situated in a text so as to indicate one of a variety of event-event relationships, such as those shown in Table 5.

	Sentences	Realis	Type	Hopper?
Pair 1	"Attacks are unavoidable." "The base was attacked."	GENERIC ACTUAL	Instance	No
Pair 2	"John will attack tomorrow." "John attacked as expected."	FUTURE ACTUAL	Completion	Yes
Pair 3	"Bill met with me." "No, he didn't."	ACTUAL NEGATE	Disagreement	Yes

Table 5: Example pairs for different realis with indications as to the specified relationship and whether they should be placed in the same hopper.

With this in mind, we have experimented with two modes for assessing realis compatibility: (1) preventing GENERIC mentions from being in the same hopper as any other type [BASIC], and (2) further preventing ACTUAL and OTHER mentions from being placed in the same hopper except for *future tense* subtypes of OTHER [STRICT]. We report the results in Table 6 along with an All-Event baseline that ignores realis.

The recall scores indicate that around 4.4% of all paired event mentions in the gold standard data represent either a GENERIC-OTHER or GENERIC-ACTUAL pairing. Furthermore, 5.9% of gold standard pairs represent relations between ACTUAL-OTHER mention pairs (excluding future tense). While preventing such pairings results in a slight reduction to recall, we believe that the accompanying gain in precision (and the improvement to the overall score) suggests that filtering out such pairs is still appropriate. Based on these observations, experiments in subsequent sections will assume that the realis module is in [STRICT] mode.

Method	Task 2			Task 3		
	PairP	PairR	KBP	PairP	PairR	KBP
Singleton Baseline	0.0%	0.0%	37.96	0.0%	0.0%	48.85
R=STRICT	23.6%	63.3%	**50.69**	30.4%	88.8%	**66.30**
R=BASIC	19.3%	65.0%	48.65	22.9%	94.7%	61.24
All Events	15.6%	65.6%	46.65	18.3%	99.1%	57.52

Table 6: Results of varying the mode in the realis compatibility module, given that event types are compatible.

6.3 Trigger Compatibility Module

If the event types and their realis labels are compatible, this module makes a binary decision on the compatibility of their event trigger words. To this end, we have experimented with six categories of trigger compatibility, each of which also accepts pairs that satisfy the stricter categories. A description of these categories along with our experimental results can be found in Table 7.

The results in Table 7 show that over 50% of all trigger pairs in the gold hoppers had no direct semantic relation in WordNet, but around 30% consisted of exact lexical matches. This suggests that any attempt at maintaining a high recall will require a source of knowledge outside of WordNet — in our case, learned equivalence sets — to determine trigger-level compatibility. As shown, we are able

Method	Description	Task 2			Task 3		
		PairP	PairR	KBP	PairP	PairR	KBP
Singleton Baseline	Baseline	0.0%	0.0%	37.96	0.0%	0.0%	48.85
T=EXACT	Triggers exactly the same	39.6%	27.0%	54.72	57.0%	29.1%	69.36
T=SAME_STEM	Triggers have same stem	35.6%	34.7%	55.69	52.4%	41.5%	72.01
T=SYNONYM	Triggers found in same synset	35.3%	38.2%	**56.59**	50.2%	47.4%	72.58
T=HYP*NYM	Triggers occur in a hypernym or hyponym relation	31.7%	40.0%	56.42	49.9%	49.5%	72.13
T=MANUAL	Triggers occur together in manually crafted equivalence set	27.1%	58.2%	55.44	38.0%	76.8%	**73.44**
T=ALL	Any two triggers can occur in the same hopper	23.6%	63.3%	50.69	30.4%	88.8%	66.30
All Events	Baseline	15.6%	65.6%	46.65	18.3%	99.1%	57.52

Table 7: Results of varying the mode of the trigger compatibility module, given that the event types and realis are compatible (R=STRICT). For SAME_STEM mode, we use a Porter stemmer, and for SYNONYM/HYP∗NYM modes, we use WordNet 3.0.

to improve our pairwise recall by 18% and 27% over a WordNet-based compatibility module for system-based and gold mentions, respectively. For this reason, experiments in subsequent sections will assume that the trigger module is in [MANUAL] mode.

6.4 Argument Compatibility Module

While event type, realis, and trigger compatibility are necessary conditions for inclusion in the same event hopper, they are not sufficient. In order to determine that two seemingly compatible event mentions genuinely represent the same event, it is also necessary to consider their arguments. In addition, different types of arguments need to be handled differently. For example, by definition, an event hopper represents a single event with a particular spatiotemporal context; thus, argument compatibility of the spatiotemporal context is required for inclusion into an event hopper. To this end, we use separate submodules for determining the compatibility of temporal arguments (temporal sub-module), spatial arguments (spatial sub-module), and all other arguments (general sub-module). In order to detect event arguments, we utilize a semantic parser and the argument extraction techniques described in Section 4.5.

First, temporal arguments are extracted and converted into a range representing the earliest possible start time and the latest possible end time associated with the event mention. The temporal ranges associated with the two mentions are then compared in the temporal sub-module to determine if there is any overlap. For instance, "over the summer" is broader than "in July", but the two ranges overlap, and for that reason, they are deemed compatible.

Second, spatial arguments are extracted and linked to a hierarchical gazetteer[3] of place names in the spatial sub-module with three possible results. If both arguments successfully link to the gazetteer, they may be found to be either (1) compatible (e.g., "France" and "Lyon") or (2) incompatible (e.g., "France" and "Italy"). If (3) one or both arguments cannot be linked to a known place name — e.g., "the church on the corner" — the pair is instead handled by the general sub-module.

Third, extracted arguments (of the same semantic role) are tested for agreement, utilizing five different techniques as shown in Table 8 as part of the general sub-module. These techniques detect matching arguments with a high precision, but lack sufficient recall.

Submodule	Match Type	Description
Lexical Match	Lex:Exact	Exact lexical match
	Lex:Head	Exact lexical match of arg heads
	Lex:Partial	One argument text contains the other
Coreference	Coref:Head	Coref chain linking the heads
	Coref:Any	Coref chain between any text in the args
WordNet	Syn:Any	Heads share any WordNet synsets
	Syn:WSD	Heads share WordNet synsets after WSD
	Hyp:Any	Heads have any hyper/hyponym relation
	Hyp:WSD	Heads have hyp* relation after WSD
Numbers	Num:Match	Both heads are the same number
	Num:Both	Heads are different numbers
	Num:One	Only one head is a number
Entity Type	Match:NOM	Head match entity type (one is nominal)
	Match:NAM	Head match entity type (both named)

Table 8: Argument compatibility submodules with the names and descriptions of particular match types.

We experimented with three modes for detecting argument compatibility, requiring arguments to match according to the particular match types in Table 8. However, the requirement that there be some argument agreement between two mentions in order

[3] from http://geonames.nga.mil/gns/html/

to include them in the same hopper is, in practice, too restrictive. Often, there are mentions in the same hopper that share no arguments in common. For this reason, we experimented with three more permissive modes of operation which do not require any arguments to match, but rather penalize argument mismatches. The impact of this module (along with descriptions of each mode) can be seen in Table 9.

Based on the results of this table, we observe that less than 20% of pairs within our hoppers have any argument match detectable by our modules, and that around half of those that do should not be placed in the same hopper due to some mismatch associated with (1) another pair of arguments, (2) the triggers, or (3) the realis. Likewise, over 20% of gold pairs in the same hopper have some detected argument mismatch. Furthermore, it appears that over 30% of all gold pairs have neither a detectable match nor a detectable mismatch. Taken together, these results suggest that a sizable number of mentions in each text do not provide the argument-level evidence needed to determine event hopper coreference using the modules described thus far. This suggests that an additional module capable of analyzing discourse patterns is required to extract explicit and implicit cues for assigning such mentions to an event hopper.

6.5 Discourse Processing Module

This final module serves to alleviate some of the difficulties associated with argument-based event hopper coreference by defining a discourse model of event coreference. This module is based on the insight that repeated mentions of the same event will follow the Gricean Maxim of Quantity and will not duplicate all of the information each time. This module provides a score for two events' compatibility.

Processing event mentions within the discourse module consists of two steps. First, we must identify the preceding mention in an event discourse chain. Then, we must determine if the adjacent mentions in the chain should be included in the same hopper, using particular discourse cues, trigger relationships, and arguments where available.

For simplicity, we define an event discourse chain as a sequence of mentions with the same trigger stem (e.g., "attacks", "attacked", "attacking", "attacker"). Then, for a given event discourse chain, we determine whether adjacent mentions in the chain should be included in the same hopper. To accomplish this, we make use of negative indicators (e.g., "unrelated", "similar", "different"), as well as positive indicators (e.g., "the", "this", "these"). Our discourse module first filters out mentions which match a negative indicator. Then, it directly detects those mentions which exhibit a positive indicator and links them to their predecessor in the discourse chain.

Table 10 shows the results for the discourse module without the argument compatibility module. We report scores assuming that the pairwise mentions in the chain have been analyzed in one of three modes: (1) all pairs are linked [ALL], (2) the pairs are linked if the later mention has a positive indicator [POSITIVE], or (3) the pairs are linked if it has a positive indicator (and no negative indicator) [POS_NO_NEG].

	Task 2			**Task 3**		
Method	PairP	PairR	KBP	PairP	PairR	KBP
Singleton Baseline	0.0%	0.0%	37.96	0.0%	0.0%	48.85
D=ALL	47.4%	27.9%	54.63	53.6%	31.3%	68.93
D=POS_NO_NEG	39.9%	5.7%	43.70	50.9%	8.5%	56.94
D=POSITIVE	39.9%	5.8%	43.78	50.7%	8.6%	57.05
D=NONE,A=ALL	27.1%	58.2%	**55.44**	38.0%	76.8%	**73.44**
All Events	15.6%	65.6%	46.65	18.3%	99.1%	57.52

Table 10: Results of varying the discourse module, given that the event types, realis, and triggers are compatible (R=STRICT, T=MANUAL).

According to our experiments, only a small percentage of pairs were linked via explicit cues (less than 9%). Furthermore, the effect of including negative cues was minimal. Overall, it remains an open research question as to how best to incorporate discourse-level information into models of within-document event coreference and event hopper construction.

6.6 Other Models

In the interest of generalizing beyond the narrow rule-based discourse module, we have also experimented with using a maximum entropy-based discourse classifier (Discourse ML), which uses as features the positive and negative indicators described in Section 6.5, as well as features for realis mismatch, argument role presence or absence, argument match type, and trigger compatibility type.

Method	Description	Task 2			Task 3		
		PairP	PairR	KBP	PairP	PairR	KBP
Singleton Baseline	Baseline	0.0%	0.0%	37.96	0.0%	0.0%	48.85
A=REQ_HIGH	Require Lex:Exact, Lex:Head, Coref:Head, WN:Syn:WSD, or Num:Match	63.3%	10.6%	49.50	68.1%	12.0%	61.89
A=REQ_MED	+ WN:Syn:Any, WN:Hyp:WSD, Num:Both, or EType:Match:NOM	51.2%	13.2%	50.57	56.9%	17.1%	63.27
A=REQ_LOW	+ Lex:Partial, Coref:Any, WN:Hyp:Any, or Num:One	51.4%	14.5%	50.69	54.4%	18.0%	63.20
A=NO_MISS	Join events if no arg mismatches	30.7%	42.4%	**55.89**	47.3%	54.9%	73.18
A=SPACE_TIME	Join events if no space/time arg mismatches	26.1%	52.1%	55.53	38.5%	73.1%	73.25
A=NO_MULTI	Join events if less than 2 argument mismatches	26.5%	55.0%	55.35	38.2%	73.7%	73.33
A=ALL	Always join events (ignore arguments)	27.1%	58.2%	55.44	38.0%	76.8%	**73.44**
All Events	Baseline	15.6%	65.6%	46.65	18.3%	99.1%	57.52

Table 9: Results of varying the argument compatibility module, given that the event types, realis, and triggers are compatible (R=STRICT, T=MANUAL).

More generally still, we have also experimented with a general machine learning approach (Full ML) that determines whether a given pair of event mentions should be combined into the same hopper. Full ML utilizes a maximum entropy classifier with the same features, but considers all pairs of the same event type, whereas the discourse model considers only adjacent pairs in a discourse chain.

Finally, we have taken into account the relative confidence of our trigger compatibility and argument compatibility modules to produce a tiered approach to event hopper combination (Tiered Model). In particular, for a given trigger compatibility rating, we require a given argument compatibility rating based on three tiers of trigger compatibility, as described in Algorithm 1.

Algorithm 1 Tiered Compatibility Algorithm

```
if T=EXACT,SAME_STEM,SYNONYM then
    A=NO_MULTI
else if T=MANUAL then
    A=M_LEARNING
else
    A=REQ_STRICT
end if
```

Results for the machine learning models — argument-level, discourse-level, and both together (Combined ML) — are compared against one another in Table 11. The tiered approach, both with and without a discourse-level ML model, is included as well. For all ML models, we compare the model (1) with no realis/trigger filtering and (2) with the optimal realis/trigger filtering (R=STRICT, T=MANUAL).

Results for the event hopper system — for Task 2 with system-provided events and realis labels, and for Task 3 with gold-standard events and labels — can be found in Sections 7.2 and 7.3.

Method	Task 2			Task 3		
	PairP	PairR	KBP	PairP	PairR	KBP
Singleton Baseline	0.0%	0.0%	37.96	0.0%	0.0%	48.85
Full ML	34.7%	36.5%	54.09	45.9%	40.4%	68.92
with best R/T	38.6%	36.1%	55.17	50.3%	39.1%	70.42
Discourse ML	35.8%	35.4%	52.36	42.6%	43.8%	67.11
with best R/T	48.9%	30.4%	54.87	59.5%	35.4%	70.05
Combined ML	39.2%	31.2%	53.57	50.2%	37.2%	68.28
with best R/T	45.8%	31.8%	55.22	54.4%	36.3%	70.20
Tiered Model	32.3%	44.2%	**55.54**	44.4%	53.8%	71.78
with Discourse ML	35.4%	36.1%	53.50	48.1%	47.1%	69.96
Best R/T	27.1%	58.2%	55.44	38.0%	76.8%	**73.44**
All Events	15.6%	65.6%	46.65	18.3%	99.1%	57.52

Table 11: Results of the machine learning models and the tiered model, separate and in conjunction. The phrase "best R/T" indicates that the system uses modes R=STRICT and T=MANUAL.

7 Evaluation

In this section, we describe our system's performance on the KBP evaluation. For each task, we submitted several runs to test different parameters of the system.

7.1 Task 1: Event Detection

For Task 1, we submitted runs using two different strategies for event trigger detection. Our first run utilized our event detection strategy that made use of 60% of the available data sets (*trainEN*) to determine prior lexical probabilities, while the second run made use of the full dataset (*allEN*). This variation serves to test the impact of additional training data on the system. The results are shown in Table 12. The *Event* metric corresponds to the KBP *mention_type* metric, which is the

micro-average F-measure over all of the event types for detecting triggers with the correct event type. The *Event+Realis* metric is equivalent to the *mention_type+realis_status*, which additionally requires the trigger and event type to have the correct realis label. Note that the Rank1 and median scores for the different metrics do not necessarily come from the same system.

	Event			Event+Realis		
Method	P	R	F	P	R	F
Rank1	—	—	58.41	—	—	44.24
LCC2	73.95	46.61	**57.18**	49.22	31.02	**38.06**
LCC1	72.92	45.91	56.35	48.92	30.81	37.81
Median	—	—	48.79	—	—	34.78

Table 12: Results of our generated event triggers, types, and realis labels.

7.2 Task 2: Event Detection and Coreference

Task 2 measures the performance of the event hopper system on top of events produced by the event detection system. We submitted three runs for Task 2, each of which utilized our event detection and realis labeling systems as input. Each of the three submissions (with scores) is indicated in Table 13, along with the median score and the score of the top-performing system.[4] LCC1 and LCC2 each utilized *allEN* for the Event Detection task and differ only in the configuration of the hopper system, while LCC3 utilized only *trainEN* to train the event detection component.

Method	**KBP Score**
Rank1	63.23
LCC2	**62.95**
LCC1	62.80
LCC3	62.63
Median	54.62

Table 13: Results of our generated event triggers, types, realis labels, and hoppers.

7.3 Task 3: Event Coreference

Task 3 specifically measures the performance of the event hopper systems. To this end, the TAC organizers provided all systems with event triggers, along with their event types and realis information. We also submitted three runs for Task 3, which are shown in Table 14, along with the median score and top-performing score for this task, which in this case was LCC3.[5] Due to the different characteristics of system events/realis and gold events/realis, these models are significantly different from those submitted as part of Task 2.

Method	**System**	**KBP Score**
Rank1	Rule-Based	**75.69**
LCC3	Rule-Based	**75.69**
LCC2	Tiered	74.87
LCC1	Machine Learning	71.86
Median	—	71.31

Table 14: Results of our submitted hopper system using provided event triggers, types, and realis labels.

8 Conclusion

In this paper, we presented a system that can successfully detect and extract events with the end goal of populating a knowledge base. Each event mention is associated with its trigger in text, provided with an event type and a realis label, and linked to other event mentions in the same document to produce an event "hopper". This system exemplifies one method for achieving a much richer knowledge base of events.

9 Acknowledgements

This work was sponsored in part by the Air Force Research Laboratory (AFRL).

[4] LCC1 and LCC3 utilized R=GENERIC, T=SYNONYM, D=POS_NO_NEG, A=SPACE_TIME, while LCC2 utilized R=GENERIC, T=TIERED, D=POSITIVE, A=TIERED.

[5] Note that LCC1 represents our best-performing machine learning model (R:STRICT, T:MANUAL, A:ML, D:ML:Type), LCC2 represents our best-performing tiered model (R:STRICT, T:TIERED, D:ALL:Syn, A:TIERED), and LCC3 represents our best-performing rule-based system (R:STRICT, T:MANUAL, D:ALL:Type, A:ALL).

References

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, Uppsala, Sweden, July. Association for Computational Linguistics.

David B. Bracewell, David Hinote, and Sean Monahan. 2014. The Author Perspective Model for Classifying Deontic Modality in Events. In *The Twenty-Seventh International Flairs Conference*.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2012. Did It Happen? The Pragmatic Complexity of Veridicality Assessment. In *Computational Linguistics*, volume 38(2), pages 301–333.

Frederik Hogenboom, Flavius Frasincar, Uzay Kaymak, and Franciska de Jong. 2011. An Overview of Event Extraction from Text. In *Proceedings of Derive2011 Workshop*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *arXiv preprint arXiv:1405.4053*.

Linguistic Data Consortium (LDC). 2005. ACE (Automatic Content Extraction) English Annotation Guidelines for Events Version 5.4.3 2005.07.01.

Linguistic Data Consortium (LDC). 2014a. DEFT Rich ERE Annotation Guidelines: Events V2.9.

Linguistic Data Consortium (LDC). 2014b. LDC2014E121 DEFT Event Nugget Evaluation Training Data.

Linguistic Data Consortium (LDC). 2015a. LDC2015E22 TAC KBP English Event Argument Extraction Comprehensive Pilot and Evaluation Data 2014.

Linguistic Data Consortium (LDC). 2015b. LDC2015E29 DEFT Rich ERE English Training Annotation V2.

Linguistic Data Consortium (LDC). 2015c. LDC2015E73 TAC KBP 2015 Event Nugget Training Data Annotation V2.

Zhengzhong Liu, Jun Araki, Eduard Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. 1998. NOMLEX: A Lexicon of Nominalizations. In *Proceedings of EURALEX*, volume 98, pages 187–193. Citeseer.

Thomas A. Mathew and E. Graham Katz. 2009. Supervised Categorization for Habitual Versus Episodic Sentences. Dissertation. Georgetown University.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *In Proceedings of Workshop at ICLR*.

Teruko Mitamura, Yukari Yamakawa, Susan Holm, Zhiyi Song, Ann Bies, Seth Kulick, and Stephanie Strassel. 2015. Event Nugget Annotation: Processes and Issues. In *The 2015 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT 2015) . 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*.

Sean Monahan and Mary Brunson. 2014. Qualities of Eventiveness. In *ACL 2014*.

Sean Monahan, Dean Carpenter, Maxim Gorelkin, Kevin Crosby, and Mary Brunson. 2014. Populating a Knowledge Base with Entities and Events. In *In Proc. Text Analysis Conference (TAC2014)*.

Roser Morante and Eduardo Blanco. 2012. *SEM 2012 Shared Task: Resolving the Scope and Focus of Negation. In *First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 265–274.

National Institute of Standards and Technology (NIST). 2014. TAC KBP 2014 Event Argument Track. `http://www.nist.gov/tac/2014/KBP/Event/`. Accessed: 2015-10-14.

National Institute of Standards and Technology (NIST). 2015. TAC KBP 2015 Event Track. `http://www.nist.gov/tac/2015/KBP/Event/index.html`. Accessed: 2015-10-15.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora.

Nils Reiter and Anette Frank. 2010. Identifying generic noun phrases. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Roser Saurí and James Pustejovsky. 2009. FactBank: a corpus annotated with event factuality. In *Language Resources and Evaluation*, volume 43 Issue 3, pages 227–268.