Automatic extraction of personal events from dialogue

Joshua D. Eisenberg
Artie, Inc.
601 W 5th St, Los Angeles, CA 90071
joshua.eisenberg@artie.com

Michael Sheriff
Florida International University
Department of Creative Writing
11101 S.W. 13 ST., Miami, FL 33199
msher043@fiu.edu

Abstract

In this paper we introduce the problem of extracting events from dialogue. Previous work on event extraction focused on newswire, however we are interested in extracting events from spoken dialogue. To ground this study, we annotated dialogue transcripts from fourteen episodes of the podcast *This American Life*. This corpus contains 1,038 utterances, made up of 16,962 tokens, of which 3,664 represent events. The agreement for this corpus has a Cohen's κ of 0.83. We have open sourced this corpus for the NLP community. With this corpus in hand, we trained support vector machines (SVM) to correctly classify these phenomena with 0.68 F1, when using episode-fold cross-validation. This is nearly 100% higher F1 than the baseline classifier. The SVM models achieved performance of over 0.75 F1 on some testing folds. We report the results for SVM classifiers trained with four different types of features (verb classes, part of speech tags, named entities, and semantic role labels), and different machine learning protocols (under-sampling and trigram context). This work is grounded in narratology and computational models of narrative. It is useful for extracting events, plot, and story content from spoken dialogue.

1 Motivation

People communicate using stories. A simple definition of *story* is a series of events arranged over time. A typical story has at least one plot and at least one character. When people speak to one another, we tell stories and reference events using unique discourse. The purpose of this research is to gain better understanding of the events people reference when they speak, effectively enabling further knowledge of how people tell stories and communicate.

There has been no work, to our knowledge, about event extraction from transcripts of spoken language. The most popular corpora annotated for events all come from the domain of newswire (Pustejovsky et al., 2003b; Minard et al., 2016). Our work begins to fill that gap. We have open sourced the gold-standard annotated corpus of events from dialogue.[1] For brevity, we will hearby refer to this corpus as the Personal Events in Dialogue Corpus (PEDC). We detailed the feature extraction pipelines, and the support vector machine (SVM) learning protocols for the automatic extraction of events from dialogue. Using this information, as well as the corpus we have released, anyone interested in extracting events from dialogue can proceed where we have left off.

One may ask: why is it important to annotate a corpus of dialogue for events? It is necessary because dialogue is distinct from other types of discourse. We claim that spoken dialogue, as a type of discourse, is especially different than newswire. We justify this claim by evaluating the distribution of narrative point of view (POV) and diegesis in the PEDC and a common newswire corpus. POV distinguishes whether a narrator tells a story in a personal or impersonal manner, and diegesis is whether the narrator is involved in the events of the story they tell. We use POV and diegesis to make our comparisions because they give information about the narrator, and their relationship to the story they tell.

We back our claim (that dialogue is different than newswire) by comparing the distributions of narrative point of view (POV) and diegesis of the narrators in PEDC with the Reuters-21578 newswire corpus.[2] Eisenberg and Finlayson (2016) found that narrators in newswire texts from the Reuters-21,578 corpus use the first-person POV less than 1% of the time, and are homodiegetic less than 1%

[1] http://www.artie.com/data/personaleventsindialogue/

[2] http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection

of the time. However, in the 14 episodes (1,028 utterances) of *This American Life*, we found that 56% narrators were first-person, and 32% narrators were homodiegetic.

We found these distributions in PEDC by using the automatic POV and diegesis extractors from Eisenberg and Finlayson (2016), which were open sourced.[3] Comparing the distributions of POV and diegesis for the PEDC to that of newswire demonstrates how different spoken dialogue is. This shows why building an annotated corpus specifically for event extraction of dialogue was necessary.

It is substantial that so many of the utterances in the PEDC are first-person narrators and homodiegetic. This means that people are speaking about their lives. They are retelling stories. They are speaking in more personal ways than narrators do in newswire. This is where the *Personal* in the Personal Events in Dialogue Corpus comes from. Additionally, using the modifier *personal* aligns this work with Gordon and Swanson (2009) who extracted personal stories from blog posts. We want our work to help researchers studying computation models of narrative.

1.1 What are *personal events*?

We define event as: an action or state of being depicted in text span. Actions are things that happen, most typically processes that can be observed visually. A state of being portrays the details of a situation, like the emotional and physical states of a character. For our work, we are only concerned with the state of being for animate objects. We use the concept of *animacy* from Jahan et al. (2018), which is defined as:

> Animacy is the characteristic of being able to independently carry out actions (e.g., movement, communication, etc.). For example, a person or a bird is animate because they move or communicate under their own power. On the other hand, a chair or a book is inanimate because they do not perform any kind of independent action.

We only annotated states of being for animate objects (i.e. *beings*) because we are interested in extracting the information most closely coupled with people or characters. We were less concerned with extracting details about inanimate objects, like the states of being in this example,"The mountain was covered with trees," and more concerned with extracting states of being describing people, like in this example, "I was so excited when the dough rose," where *excited* is a state of being describing the speaker.

In the prior section we showed the PEDC contains a significant amount of personal events by running the POV and diegesis extractors from Eisenberg and Finlayson (2016). We found that the PEDC contains 56% first-person narrators, and 32% homodiegetic narrators. Our corpus has a significant amount of narrators telling personal stories.

1.2 Outline

First, in §2 we discuss the annotation study we conducted on fourteen episodes of *This American Life*. Next, in §3 we discuss the design of the event extractor. In §3.2 we discuss the different sets of features extracted from utterances. In §3.2 we talk about the protocols followed for training of support vector machine (SVM) models to extract events from utterances. In §4 we discuss the types of experiments we ran, and present a table containing the results of 57 experiments. The goal of these experiments is to determine the best set of features and learning protocols for training a SVM to extract events from dialogue. In §5 we discuss the results. In §6 we sumarize our contributions.

2 Personal events in dialogue annotation study

When beginning to think about extracting events from dialogue, we realized there is no corpus of transcribed dialogue annotated for events. There are many corpora of other text types with event annotations. TimeBank contains newswire text (Pustejovsky et al., 2003b). MEANTIME is made up of Wikinews articles (Minard et al., 2016).

Additionally there are many event annotation schema; one of the more prominent ones is TimeML (Pustejovsky et al., 2003a). We decided to develop our own annotation scheme due to the complexity of TimeML; it's an extremly fine-grained annotation scheme, with specific tags for different types of events, temporal expressions and links. We decided it would be too difficult to use TimeML while maintaining a high inter-annotator agreement and finishing the annotation study in a short amount

[3] https://dspace.mit.edu/handle/1721.1/105279

Episode Number	Episode Name	Utterances	Event tokens	Nonevent tokens	Cohen's Kappa
608	The Revolution Starts at Noon	50	130	630	0.8900
610	Grand Gesture	151	494	2032	0.7982
617	Fermi's Paradox	82	271	1204	0.8258
620	To Be Real	75	156	756	0.8258
621	Who You Gonna Call	49	104	435	0.8486
625	Essay B	54	130	417	0.8446
627	Suitable for Children	28	80	322	0.8362
629	Expect Delays	44	125	391	0.8320
639	In Dog We Trust	43	183	651	0.8777
647	LaDonna	51	143	477	0.8600
650	Change You Can Maybe Believe In	87	420	1629	0.8193
651	If You Build It Will They Come	64	264	880	0.8344
655	The Not So Great Unknown	89	400	1164	0.8256
691	Gardens of Branching Paths	171	764	2310	0.8302
Totals		1,038	3,664	13,298	
Average Kappa					0.8320

Table 1: Statistics for event annotations in dialogue corpus

of time (three months), and within a modest budget.

Given that our goal was to understand spoken conversational dialogue, we decided to create a corpus from transcribed audio. This matches the nature of the data we intend to use for our event exctractor: audio recordings of dialogue that have been transcribed as a text file.

We weighed a number of different sources for the text transcripts, but we ultimately decided to use transcripts from the podcast *This American Life*[4]. We chose this podcast because: 1) The transcripts are freely available online. 2) A significant portion of these podcasts are made up of conversations, as opposed to narration. Additionally, *This American Life* formats their transcripts so that the conversations are indented as block quotations. This made it easy to separate conversations from typical podcast narration. 3) The subject matter of *This American Life* are typically stories from people's lives. We wanted our corpus to be made up of unscripted conversations; contemporary everyday conversations, so that the extractors we train from this data are better suited to understanding people talking about their lives.

2.1 Annotation study procedures

The two authors of this paper were the annotators for this study. The first author wrote the annotation guide[5]. We trained by reading the first version of the guide, discussing short-comings, and then compiling a new version of the guide. Next, we both annotated episode 685[6]. Since we were training, we were allowed to discuss questions regarding annotation decisions. After we both finished, we ran the annotations through a program that found all the utterances with disagreements, and we discussed the mistakes.

After adjudicating the training episode, the first author updated the annotation guide to address inconsistencies we found during adjudication. Next, we began the actual annotation study. While annotating each episode, we could not discuss specifics about the utterances. We independently annotated each episode.

Once both annotators finished their annotations for an episode, we used a program we made that compared the annotations for each utterance. If there was any disagreement between the two annotators, both sets of markings from the annotators were added to an adjudication list. Then, we went through each utterance with disagreements, and discussed how the markings should be corrected for the gold-standard. We adjudicated each episode before annotating the next so that we, as annotators, could learn from each other's mistakes. Once

[4]https://www.thisamericanlife.org/

[5]http://www.artie.com/data/personaleventsindialogue/

[6]https://www.thisamericanlife.org/685/transcript

the correction lists were created, they were used along with the original markings to create the gold-standard.

2.2 Annotation syntax

Before we discuss the annotation syntax, please take a look at an annotated utterance from episode 650[7]:

```
Alan:  Due to safety
{concerns}, safety
{purposes}.  But I mean,
I can {type out} a little
bit of, like, whatever
you {want} to {tell} them,
{tell} the shelter, and I
can {make sure} they {get}
the {message} if that'll
{work} for you.
```

The annotations were marked in text files. Each text file contains an episode transcript formated so that each utterance was on its own line. The spans of text that an annotator considered events were surrounded with brackets. Usually events were single words, but occasiaonally events were multi-word expressions, like the phrase *type out* above. For more information about what we considered an event, and which state-of-beings were considered events, please refer to our annotation guide[8].

2.3 Inter-annotator agreement

We used the Cohen's Kappa metric (κ) to compute the inter-annotator agreement Landis and Koch (1977). According to Viera et al. (2005) κ values above 0.81 are considered *almost perfect agreement*. The average κ for our annotations is 0.83 so our inter-annotator agreement is *almost perfect*. This average κ is a weighted average, where the κ for each episode is multiplied by the number of utterances in the episode. Once the sum of weighted averages is obtained, we divide by the total number of utterances in the corpus.

The κ for event extraction measures inter-annotator agreement for a binary classification task for each token across each utterance. If both annotators marked a token as an event, this counted as a true positive, and if both annotators marked a token as a non-event, this is counted as a true negative. All other cases are disagreements; these were adjudicated by both authors. A token can be annotated as an *event*, or a *non-event*.

3 Developing the extractor

Our extractor was implemented in Java. This is due to the availability of high-quality open-sourced NLP libraries. There are two aspects of the extractor's design that we will cover: 1) feature engineering and 2) protocols for training SVM models.

3.1 Feature engineering

First, we will discuss the different types of features that we extracted from each utterance in the corpus.

3.1.1 Part of speech tags

We used the part of speech (POS) tagger (Toutanova and Manning, 2000; Toutanova et al., 2003) from Stanford CoreNLP (Manning et al., 2014) to extract part of speech tags for each word in each utterance of our corpus. We used the `english-bidirectional-distsim` model. This model was chosen since it has the most accurate performance, even though it has a slower run-time. For the purpose of these experiments run-time wasn't a limiting factor.

Each POS tag was assigned a unqiue integer value between 1 and 36. If a token has no POS tag, then it is assigned the value of -1. The following is the procedure for mapping POS tags into feature vectors: First, use Stanford CoreNLP to find the POS tags for each token in an utterance. Second, produce a vector of length 37 for each token, and fill each element with a -1. Third, for the vector representing each token, change the value of the element with the index cooresponding to the particular POS of the current token to 1. If the token has no POS tag, then the vector is unchanged.

3.1.2 Named entity tags

We used the Named Enitity Recognizer (NER) (Finkel et al., 2005) from Stanford CoreNLP (Manning et al., 2014) to extract named entity types from utterances. We included named entity tag as a feature type for event extraction because we hypothesized that some named entity types should never be considered as events, like `PERSON`, `ORGANIZATION`, and `MONEY`. However, the `DATE` and `DURATION` classes were often classified as events.

The NER tag feature was encoded into a vector of length nine. The first eight elements of this vector each represents one of the eight NER classes

[7] https://www.thisamericanlife.org/650/transcript
[8] http://www.artie.com/data/personaleventsindialogue/

in Stanford's NER. The vector's final element represents whether the current token is not a named entity. This is the procedure for extracting NER tag features from an utterance: First use Stanford CoreNLP to find the NER tags for each token in an utterance. Second, produce a vector of length nine for each token, and fill each element with a -1. Third, for the vector representing each token in an utterance, change the value of the element with the index corresponding to a particular NER tag of the current token to 1. If there is no NER tag for a given token, then set the final element of the vector to 1.

3.1.3 Verb classes

We use a similar pipeline as Eisenberg and Finlayson (2017) for verb class extraction. This pipeline determines which VerbNet (Schuler, 2005) verb classes each token in an utterance is represented by. A verb class is a set of verbs with the same semantics. For example, the verbs *slurp*, *chomp*, and *crunch* all belong to the verb class *chew*. We hypothesize that knowledge of what verb classes are instantiated by specific words is essential to extracting events from dialogue.

The features for verb classes are encoded into a vector of length 279. The first 278 elements represent which of the 278 verb classes is invoked by the current token. The final element represents if no verb classes are instantiated by the token. For the first 278 elements we use the following bipolar encoding: 1 if the verb class is instantiated in the token, or -1 if not. Note that any token can instantiate more than one verb class. The final element in the vector is assigned a 1 if no verb classes are represented by the current token, or -1 if verb classes are used (which means at least one of the first 278 elements has the value of 1).

Here is a quick overview of the pipeline for verb class extraction: first, we use *It Makes Sense* to perform word sense disambiguation on an utterance (Zhong and Ng, 2010). This produces WordNet sense keys (Fellbaum, 1998) for each token in an utterance. Next we use JVerbnet[9] to map WordNet sense keys to VerbNet classes. This produces a list of VerbNet classes for each token. Finally, each list is mapped to a bipolar feature vector of length 279, as explained in the paragraph above.

[9] http://projects.csail.mit.edu/jverbnet/

3.1.4 Semantic role labels

We use the *Path LSTM* Semantic Role Labeler (SRL) to extract a set of features from utterances (Roth and Lapata, 2016). We extract two features for each token in an utterance: 1) is the token a predicate? and 2) is the token an argument of a predicate? These features fill a vector of length two, and once again we use bipolar encoding as all the previous features discussed in this section.

There are many more features in Path LSTM, however we didn't have the time to find an intelligent way to use them. One of those features is the ability to parse into semantic frames from FrameNet (Baker and Sato, 2003). Path LSTM can parse into the over 1,200 semantic frames in FrameNet. We hypothesize that knowing which tokens represent different frame elements for each frame would be a useful feature extracting events from dialogue. This feature would provide even more fine-grained information than the verb class features.

Here is the way we extracted SRL features from utterances: first, for each utterance use Path LSTM to extract an SRL parse. Second, produce a feature vector of length two for each token in the utterance, and initialize both elements to -1. Third, get the list of predicates from the SRL parse. For each token, if it is a predicate, set the first element of the feature vector to 1. Otherwise, do nothing. Fourth, for each predicate, get the argument map. For each token, if it is a member of any argument map, set the second element of the feature vector to 1. Otherwise, do nothing.

3.2 Learning protocols

Second, we discuss the details about how the SVM models were trained.

3.2.1 Cross-validation

We used 14-fold cross-validation, or colloquially speaking *episode-fold cross-validation*. There are 14 episodes in our corpus. For each fold of cross-validation, one episode is reserved for testing, and the remaining 13 folds are used for training. This procedure is performed 14 times, so that each of the 14 episodes has the chance to be used as testing data.

3.2.2 Under-sampling

We incorporated under-sampling into our SVM based experiments. Undersampling is a technique for boosting performance of models when training

Features				ML	Protocols	Events			Nonevents		
POS	NER	Verb Classes	SRL	Under-sampling	Trigrams	F1	Precision	Recall	F1	Precision	Recall
X						0.5763	0.7364	0.4791	0.9095	0.8705	0.9529
X				X		0.6185	0.4777	0.8848	0.8347	0.9595	0.7390
X					X	0.5786	0.7435	0.4791	0.9104	0.8706	0.9545
X				X	X	0.6262	0.4890	0.8783	0.8420	0.9579	0.7516
	X					NaN	0	NaN	0.8805	0.7871	1.0
X	X					0.5800	0.7366	0.4840	0.9098	0.8715	0.9523
X	X			X		0.6205	0.4837	0.8768	0.8381	0.9572	0.7463
X	X				X	0.5823	0.7435	0.4841	0.9107	0.8717	0.9539
X	X			X	X	0.6252	0.4870	0.8818	0.8406	0.9588	0.7489
		X				0.5609	0.9281	0.4039	0.9206	0.8594	0.9918
		X		X		0.5637	0.9225	0.4079	0.9207	0.8601	0.9910
		X			X	0.5608	0.9044	0.4087	0.9195	0.8599	0.9886
		X		X	X	0.5658	0.6457	0.5074	0.8979	0.8729	0.9249
X		X				0.6698	0.8593	0.5530	0.9298	0.8886	0.9756
X		X		X		0.6763	0.5555	0.8712	0.8792	0.9586	0.8123
X		X			X	0.6755	0.8475	0.5654	0.9299	0.8911	0.9727
X		**X**		**X**	**X**	**0.6794**	**0.5593**	**0.8736**	**0.8805**	**0.9594**	**0.8141**
	X	X				0.5648	0.4088	0.4088	0.9209	0.8604	0.9912
	X	X		X		0.5675	0.9188	0.4126	0.9209	0.8611	0.9903
	X	X			X	0.5676	0.9188	0.4126	0.9210	0.8611	0.9903
	X	X		X	X	0.5666	0.6411	0.5116	0.8972	0.8738	0.9226
X	X	X				0.6732	0.8580	0.5579	0.9301	0.8896	0.9750
X	X	X		X		0.6736	0.5513	0.8731	0.8768	0.9589	0.8082
X	X	X			X	0.6769	0.8460	0.5679	0.9300	0.8917	0.9722
X	X	X		X	X	0.6750	0.5510	0.8792	0.8764	0.9606	0.8062
			X			0.4687	0.6751	0.3613	0.8959	0.8458	0.9530
			X	X		0.5287	0.4764	0.6000	0.8509	0.8826	0.8220
			X		X	0.4687	0.6751	0.3613	0.8959	0.8458	0.9530
			X	X	X	0.5287	0.4764	0.6000	0.8509	0.8826	0.8220
X			X			0.5763	0.7364	0.4791	0.9095	0.8705	0.9529
X			X	X		0.6239	0.4892	0.8726	0.8416	0.9559	0.7526
X			X		X	0.6564	0.7667	0.5826	0.9207	0.8923	0.9519
X			X	X	X	0.6257	0.4868	0.8833	0.8408	0.9595	0.7487
	X		X			0.4717	0.6786	0.3636	0.8965	0.8463	0.9535
	X		X	X		0.5359	0.4872	0.6017	0.8552	0.8839	0.8289
	X		X		X	0.4714	0.6784	0.3633	0.8965	0.8463	0.9535
	X		X	X	X	0.5358	0.4872	0.6015	0.8552	0.8839	0.8290
X	X		X			0.5800	0.7366	0.4840	0.9098	0.8715	0.9523
X	X		X	X		0.6150	0.4752	0.8824	0.8319	0.9587	0.7358
X	X		X		X	0.6631	0.7739	0.5897	0.9221	0.8942	0.9529
X	X		X	X	X	0.6254	0.4862	0.8847	0.8402	0.9598	0.7475
		X	X			0.5609	0.9281	0.4039	0.9206	0.8594	0.9918
		X	X	X		0.5769	0.5037	0.6812	0.8588	0.9036	0.8187
		X	X		X	0.5608	0.9044	0.4087	0.9195	0.8599	0.9886
		X	X	X	X	0.5753	0.4937	0.6977	0.8528	0.9068	0.8058
X		X	X			0.6698	0.8593	0.5530	0.9298	0.8886	0.9756
X		X	X	X		0.6759	0.5566	0.8679	0.8795	0.9575	0.8138
X		X	X		X	0.6760	0.8477	0.5660	0.9300	0.8912	0.9727
X		X	X	X	X	0.6769	0.5535	0.8780	0.8780	0.9605	0.8089
	X	X	X			0.5648	0.9244	0.4088	0.9209	0.8604	0.9912
	X	X	X	X		0.5841	0.5132	0.6845	0.8628	0.9051	0.8248
	X	X	X		X	0.5648	0.9011	0.4137	0.9198	0.8610	0.9879
	X	X	X	X	X	0.5824	0.4900	0.7243	0.8499	0.9131	0.7955
X	X	X	X			0.6732	0.8580	0.5579	0.9301	0.8896	0.9750
X	X	X	X	X		0.6732	0.5502	0.8748	0.8765	0.9595	0.8072
X	X	X	X		X	0.6769	0.8460	0.5679	0.9300	0.8917	0.9722
X	X	X	X	X	X	0.6729	0.5491	0.8764	0.8758	0.9601	0.8006

Table 2

on unbalanced datasets (Japkowicz et al., 2000). Our event corpus has about four nonevents for every one event. To mitigate this, during training a SVM model on an episode we add the feature vectors for every event to the training set. Next, we count the number of feature vectors for events in the training set. Then, we randomly select nonevent feature vectors, and add the same number of vectors to the training set as there are event vectors. Hence, for every event feature vector in the training set, there is only one nonevent feature vector. In our experiments (§4) we saw that undersampling raised the F1 for most feature sets. Our implementation of under-sampling allows us to toggle it on and off for different experiments. Hence, undersampling could be parameterized, along with the types of features used, and other variations on the SVM learning discussed below.

Since there is an element of randomness in our implementation of undersampling, we ran each undersampling experiment 100 times. We report the result for the experiment that had the highest F1 relative to the event class. This is a somewhat crude approach. In the future, we would like to employ an entropy based approach, where we select which majority class feature vectors to use based on the entropy of the set of vectors.

3.2.3 Simulating context through trigrams

We simulate context by appending feature vectors for neighboring words to the current word's feature vector. Specificially, for each token, get the feature vector for the preceding token and the feature vector for the proceeding token, and append these two vectors to the original vector. If there is no preceding token make a feature vector where each element is -1. The length of this negative vector is that of the original feature vector. Similarly, follow the same procedure if there is no proceeding token. Using trigram context vectors slightly raised the F1 for many SVM models, but it did not have a significant effect. This leads us to hypothesize that there is probably a better way to encode context for this task.

Our implementation of trigram context is modular, along with the other learning protocols: context can be toggled for any experiment. Furthermore, experiments that make use of trigram context, can also take advantage of under-sampling. Each set of features can have four seperate experiments: 1) training with no augmentations, 2) training with under-sampling, 3) training with trigram context vectors, and 4) training with both under-sampling and trigram context vectors.

3.2.4 SVM hyperparameters

All our SVM models used a linear kernel. We chose a linear kernel because of bipolar encoding of the feature values, and it produced the best F1 during early experiments. The hyperparameters for all the SVMs were as follows: $\gamma = 0.5$, $\nu = 0.5$, C = 20, and $\epsilon = 0.01$.

4 Results

We report our results in Table 2. The table is organized in four vertical columns, from left to right:

1) *Features*: this section contains the features used for an experiment. The possible types of features are POS, NER, Verb Classes, and SRL. The combination of features used for an experiment are indicated by X's in the column of the corresponding features. There are four possible experiments (for each of the four possible machine learning protocols chosen) run for a given feature type. In rare cases (like for experiments with only NER features) only the basic experiment results are reported because the SVM classifier could not adequately learn and classify everything as a nonevent.

2) *ML Protocols*: this section contains the machine learning (ML) protocols used for an experiment. The possible protocols are: undersampling and trigrams. The combination of ML protocols used for an experiment are indicated by X's in the column of the coresponding protocols. For each combination of features, four experiments are run. Each of the four experiments, for a feature set, represent a unique combination of the two ML protocols.

3) *Events*: in this section we report the results (F1, precision, and recall) for all tokens that were marked as events in the gold-standard.

4) *Nonevents*: similarly, in this section we report the results for all tokens that were marked as nonevents in the gold-standard.

Table 2 contains all combinations of features and ML protocols. We report all the results to show the fluctuations of performance for different combinations of features and protocols.

We will compare the results in Table 2 to a minority class baseline. For our experiments, the minority class is the event class. We are interested in maximizing the F1 of the event class as opposed to the nonevent class, because we want to accurately extract events. Events are more rare than nonevents, hence this is the phenomena we are exploring. Our baseline, relative to the event class is: F1 = 0.3553, precision = 0.2160, and recall = 1.

5 Discussion

Our best performing event extractor uses POS and verb class features, and the ML protocols used were undersampling and trigrams, however, the performance is not significantly better than the extractors that only use either one of the two protocols. Our best peforming event extractor with no extra ML protocols was the extractor with POS, NER, and verb class features. The performance of the extractor that had all four features had the same performance as the former, so we can say that the addition of SRL features adds no extra information to the classification process.

It is interesting to see the affect of undersampling on performance. It boosted the event F1 for most feature sets. Not only did it boost the F1, but it flipped the values of precision and recall with respect to the original experiment. Without undersampling, the precision is always higher than the recall. Once undersampling is toggled, the recall becomes larger than the precision. Also, the undersampled recall is typically higher than the non-

undersampled precision. This flippage is important to note for situations when the event extractor is actually used in real-world systems.

If the situation requires a minimal number of false positives, than precision should be maximized, therefore no undersampling should be used when training the model. However, if minimal false negatives is a bigger priority, then recall should be maximized, hence undersampling should be used in training. Whether undersampling is used, or not, depends on the actual context the event extractor is being deployed.

In general, undersampling helped boost performance of event classification in most experiments. Trigrams gave an even smaller boost to event classification in most experiments. Experiments that had both undersampling and trigrams had the largest boost when compared to the experiment with no extra ML protocols.

There were two feature sets that trigram context had a significant affect, both POS + SRL and POS + VERB + SRL. These are the only experiments where the trigram context protocol led to the greatest performance for the feature set, and by a significant margin. Overall, trigrams had a much smaller affect on overall performance. We hypothesize that there are better ways to implement this form of context. Either a classifier that's better suited for sequential data should be used, or a different form of encoding the context feature should be explored. Another note about a negative result: the impact of the SRL features was much less influential than we hypothesized. Going forward, we think that the actual semantic frames instantiated should be used as features, as well as different frame elements, and not just occurence of predicates and arguments.

6 Contributions

In this paper we presented two sets of contributions: First, we have open sourced the first corpus of dialogue annotated for events.[10] This corpus can be used by researchers interested in the automatic understanding of dialogue, specifically dialogue that is rich with the personal stories of people. Second, we share the design and evaluate the performance of 57 unique event classifiers for dialogue. These results can be used by researchers to decide which features and machine learning protocols should be implemented for their own event extractors. Our best performing extractor has a 0.68 F1, which is over 200% higher than baseline. We hope that this work can be used by the community to better understand how people reference events from stories in dialogue.

[10] http://www.artie.com/data/personaleventsindialogue/

Acknowledgments

A huge thanks to *This American Life* for giving us permission to distribute their transcripts for the open sourced PEDC. Thanks to Frances Swanson for coordinating the permissions. Thanks to Ryan Horrigan and Armando Kirwin, from Artie, Inc., for giving me the time and resources to pursue this research. Thanks to Aimee Rubensteen for being an amazing editor.

References

Collin F. Baker and Hiroaki Sato. 2003. The FrameNet data and software. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 161–164, Sapporo, Japan. Association for Computational Linguistics.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Joshua Eisenberg and Mark Finlayson. 2016. Automatic identification of narrative diegesis and point of view. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 36–46, Austin, Texas. Association for Computational Linguistics.

Joshua Eisenberg and Mark Finlayson. 2017. A simpler and more generalizable story detector using verb and character features. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2708–2715, Copenhagen, Denmark. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet: An electronic lexical database*. MIT Press.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, page 363–370, USA. Association for Computational Linguistics.

Andrew Gordon and Reid Swanson. 2009. Identifying personal stories in millions of weblog entries. In *Third International Conference on Weblogs and Social Media, Data Challenge Workshop, San Jose, CA*, volume 46, pages 16–23, San Jose, CA.

Labiba Jahan, Geeticka Chauhan, and Mark Finlayson. 2018. A new approach to Animacy detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1–12, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Nathalie Japkowicz et al. 2000. Learning from imbalanced data sets: a comparison of various strategies. In *AAAI workshop on learning from imbalanced data sets*, volume 68, pages 10–15. Menlo Park, CA.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begoña Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son. 2016. MEANTIME, the NewsReader multilingual event and time corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4417–4422, Portorož, Slovenia. European Language Resources Association (ELRA).

James Pustejovsky, José M Castano, Robert Ingria, Roser Sauri, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R Radev. 2003a. Timeml: Robust specification of event and temporal expressions in text. *New directions in question answering*, 3:28–34.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003b. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40. Lancaster, UK.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. PhD dissertation, University of Pennsylvania.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, page 63–70, USA. Association for Computational Linguistics.

Anthony J Viera, Joanne M Garrett, et al. 2005. Understanding interobserver agreement: the kappa statistic. *Fam med*, 37(5):360–363.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83, Uppsala, Sweden. Association for Computational Linguistics.