Behind the Scenes of an Evolving Event Cloze Test

Nathanael Chambers
Department of Computer Science
United States Naval Academy
nchamber@usna.edu

Abstract

This paper analyzes the narrative event cloze test and its recent evolution. The test removes one event from a document's chain of events, and systems predict the missing event. Originally proposed to evaluate learned knowledge of event scenarios (e.g., scripts and frames), most recent work now builds ngram-like language models (LM) to beat the test. This paper argues that the test has slowly/unknowingly been altered to accommodate LMs. Most notably, tests are auto-generated rather than by hand, and no effort is taken to include core script events. Recent work is not clear on evaluation goals and contains contradictory results. We implement several models, and show that the test's bias to high-frequency events explains the inconsistencies. We conclude with recommendations on how to return to the test's original intent, and offer brief suggestions on a path forward.

1 Introduction

A small but growing body of work is looking at learning real-world event knowledge. One particular area is how to induce event structures called schemas, scripts, or frames. This is a wide field, but variations on the narrative cloze test are often used to evaluate learned models. However, their current form has evolved beyond the cloze's original purpose. It has evolved into a language modeling (LM) task rather than an evaluation of knowledge. One proposal suggested avoiding the cloze test absent other options (Rudinger et al., 2015), but we argue that it can be useful if carefully formulated. This is the first paper to evaluate *why* LMs can seemingly succeed on the event cloze. This is also the first paper to reconcile contradictory results across recent papers. We reproduce several models for cloze prediction, include a new instance-based learning model, and show how high-frequency events pollute the test. We conclude by discussing the future of the cloze in regards to new corpus developments.

2 Previous Work

2.1 The Original Narrative Event Cloze

The narrative event cloze task was first proposed in Chambers and Jurafsky (2008). These papers introduced the first models that automatically induced event structures like Schankian scripts from unlabeled text. They learned chains of events that form common-sense structures. An event was defined as a verb/dependency tuple where the main entity in a story (the protagonist) filled the typed dependency of the verb. The following is an example with its corresponding event chain:

Text
The police arrested Jon but he escaped.
Jon fled the country.
Chain
(arrested, object), (escaped, subj), (fled, subj)

This is one instance of a chain. Research focuses on generalizing this knowledge to a stereotypical script of when a suspect escapes. In order to evaluate this generalized knowledge, the narrative cloze was proposed as one possible test. Given a set of known events, the test removes one, and a system must predict which was removed. Using the same short example:

(arrested, object), (escaped, subject), (____, ____)

Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics, pages 41–45,
Valencia, Spain, April 3, 2017.

A model of scripts can produce a ranked list of likely events to fill the hole. The test evaluates where in the ranking the correct event is found. Critically, these event tests were *manually* extracted from *hand-selected* documents.

2.2 Language Modeling Event Cloze

Jans et al. (2012) focused solely on the narrative cloze test as an end goal. They cited the cloze evaluation from Chambers and Jurafsky (2008), but made several cloze modifications that we argue make it more amenable to language modeling. Since then, subsequent work has adopted the Jans evoluation of the cloze test. There are three main changes to the original **Narrative cloze** that turned it into an **LM cloze**.

Automatic Tests: First, the LM cloze tests are *automatically generated* with all the mistakes of parsers and coreference. The original narrative cloze was manually created by human annotators who interpreted documents and extracted sets of events by hand. Accuracy was "true accuracy", and it tested only one central chain in each document. It is not clear why everyone switched to LM cloze, but Jans et al. (2012) is revealing, *"Rather than manually constructing a set of scripts on which to run the cloze test, we ... use the event chains from that section as the scripts for which the system must predict events."* Their version is not the narrative cloze from Chambers and Jurafsky. This change created what is often desired: a quick automated test with instant results.

Text Order: The LM cloze is an evaluation where events are *ordered* to know the text position of the missing event. The original narrative cloze did not require ordering information because document order does not imply real-world order, and scripts focused on real-world structure. This change naturally benefits text language models.

All Chains: Instead of selecting the central entity in a document and testing that scenario's chain, they included all entity chains. Different papers vary on this detail, but all appear to auto-extract multiple chains per document. Some include minimum chain length requirements.

All Events: Fourth, the LM cloze includes all repeated events in a chain. If an event chain contains 5 'said' events, 5 cloze tests with the same answer 'said' are in the evaluation. Critically, variants of 'said' make up 20% of the data. The original cloze only included unique events without repetition. It also intentionally omitted all 'be' events, avoiding another frequent/uninformative event in the tests (Chambers and Jurafsky, 2008). To clearly illustrate the problem, below is one such LM cloze test:

X criticized, X said, X distributed, X asking, X said, X said, X said, X said, X admitted, X asked

The narrative cloze would only test one *X said* instead of five. This seemingly small evaluation detail drops a unigram model's 'said' prediction from 50% (5 of 10) to 17% (1 of 6) accuracy.

Subsequent work adopted these changes. Pichotta and Mooney (2014) proposed a multi-argument language model. They showed that bigrams which take into account all entity arguments can outperform bigrams that only use a single argument. Rudinger et al. (2015) showed that a log-bilinear language model outperforms bigram models on the same LM cloze. Several have proposed neural network LMs (Pichotta and Mooney, 2016; Modi, 2016). Granroth-Wilding and Clark (2016) made the cloze a multiple choice prediction rather than a ranking. Curiously, they auto-generated *one* chain per document instead of all chains, and required that chain to be at least 9 events in length. Ahrendt and Demberg (2016) build on the n-gram models with argument typing and use the cloze test on a non-news corpus.

Notably with these variations, results across papers contradict. A frequency baseline is the best in some, but not in others. A PMI-based counting approach is poor in some, but close to state-of-the-art in others. Rudinger et al.'s best LM leads them to conclude that either (1) script induction should use LMs, or (2) the cloze should be abandoned. We argue instead for a third option: the LM cloze should find its way back to the original intent.

3 Data Processing

To be consistent with recent work, we use the English Gigaword Corpus for training and test. We parse into typed dependencies with the CoreNLP toolkit, run coreference, and extract event chains connected by a single entity's mentions. Each coreference entity then extracts its event chain, made up of the predicates in which it is a subject, object, or preposition argument. An event is a tuple similar to Pichotta and Mooney (2014): (s, o, p, event) where s/o/p are the subject, object, and preposition unique entity IDs. Entity singletons

are ignored, and *all* chains of length 2 or above are extracted as in these recent works.

4 Models

In order to ground our argument in the correct context, we implemented the main models from Chambers and Jurafsky (Chambers and Jurafsky, 2008), Jans et al. (2012), and Pichotta and Mooney (2014). Others have been proposed, but these core models are sufficient to illustrate the idiosyncrasies shared by LMs on event cloze prediction.

4.1 Unigrams

The unigram model is based on frequency counts from training. We define a similarity score for an event e in a chain of events c at insertion index k:

$$sim_u(e, c, k) = C(e)/N \quad (1)$$

where $C(e)$ is the count of event e and N is the number of events seen in training.

4.2 Bigrams

The bigram model is formulated as an ordered text equation as in Jans et al. (2012):

$$sim_b(e, c, k) = \prod_{i=0}^{k} P(e|c_i) * \prod_{i=k+1}^{n} P(c_i|e) \quad (2)$$

where the conditional probability is defined:

$$P(x|y) = \frac{C(x, y) + \lambda}{C(y) + |E| * \lambda} \quad (3)$$

where $C(x, y)$ is the text ordered bigram count, E is the set of events, and λ a smoothing parameter.

4.3 PMI

Pointwise mutual information was the central component of Chambers and Jurafsky (2008). They learned a variety of script/event knowledge including argument type information that is not necessarily evaluated in the LM cloze. However, for consistency, previous work tends to duplicate their prediction model as follows:

$$sim_p(e, c, k) = \sum_{i=0}^{n} log \frac{P(c_i, e)}{P(c_i)P(e)} \quad (4)$$

where the joint probability is defined:

$$P(x, y) = \frac{C(x, y) + C(y, x)}{\sum_i \sum_j C(e_i, e_j)} \quad (5)$$

Jans et al. (2012) propose an *ordered* PMI that we omit for simplicity. They found that ordering doesn't affect PMI (but is required for bigrams).

4.4 Multi-Argument N-Gram Models

The above models use a single entity in a chain (arrested X, X escaped, X fled). Pichotta and Mooney (2014) explored richer models that consider all arguments with the events. The single chain now becomes (Y arrested X, X escaped, X fled Z). If other entities are repeated across events, it uses the same variable/ID so that coreference can be modeled beyond the main entity. The n-gram models are slightly more complicated now that arguments need to be normalized, particularly in how events are counted and how the conditional probability is computed. We refer the reader to their paper for a complete formulation.

This richer formulation has not been adopted by later work, possibly due to its complexity, but we duplicated their models for completeness.

4.5 Instance-Based N-Grams

We also propose a novel extension to previous work in an attempt to not just duplicate performance, but maximize its results. Instead of training on all documents, we train *on-the-fly* with an instance-based learning approach. Given a chain of events, the algorithm retrieves documents in Gigaword that contain all the events, and computes counts $C(x)$ and $C(x, y)$ only from that subset of documents. A parameter can be tuned to require X% of the chain events to match in a document. We duplicated both unigrams and bigrams (as above) with this on-the-fly training method.

5 Experiment Setup

There are two ways to evaluate event prediction with scripts. The first is to follow a single actor through a chain of events, and predict the missing link in the chain. This prediction ignores other event arguments and only evaluates whether the system predicts the predicate and the correct syntactic position of the entity. This was part of the original *narrative cloze* from Chambers and Jurafsky (2008). The example in Section 2 illustrates such a chain. Pichotta and Mooney (Pichotta and Mooney, 2014) proposed a richer test that requires all arguments of the missing event. A single actor is still tracked through a chain of events, but correct prediction requires the complete event.

We trained on 12.5 million AP documents from Gigaword with duplicates removed. The test set is 1000 random event chains not in training. Parameters were tuned on a smaller set of dev documents.

Single Argument Chains

Model	Recall@50
Unigrams	0.338
Uni Exact 100%	0.347
Uni Exact 50%	0.386
Bigrams (k=2)	0.465
Bi Exact 100% (k=2)	0.460
PMI	0.038
PMI w/cutoff	0.391

Table 1: Single entity event chain results.

Multiple Argument Chains

Model	Recall@50
Unigrams	0.322
Uni Exact 100%	0.332
Uni Exact 50%	0.368
Bigrams (k=5)	0.408
Bi Exact 100% (k=5)	0.396
PMI	0.068
PMI w/cutoff	0.364

Table 2: Multiple argument event chain results.

6 Results

Table 1 shows model performance. The best unigram model used our new instance-based learning, but bigrams gain by 8% absolute. Notably, **PMI** performs poorly as in Jans et al. (2012) and Rudinger et al. (2015). However, by adding a frequency cutoff, *the poor result is reversed*. Figure 1 shows the cutoff recall curve. Both papers concluded that PMI was the problem, but we found it is simply the *over*-evaluation of frequent events.

PMI is known to prefer infrequent events, and this is evident by looking at the *information content* (IC) of model predictions. The information content of an event is its log probability in the Gigaword Corpus. What types of events do language models predict? Table 3 shows that the average LM prediction contains far less information. Perhaps more clear, Table 4 shows an actual list of predictions for one cloze test. The n-gram models predict frequent events, but PMI predicts seemingly more meaningful events. We are not arguing in favor of PMI as a model, but simply illustrating how frequency explains almost all of the contradictions in previous work.

Finally, Table 2 mirrors the relative results of single arguments in the multi-argument setting. Once again, a simple cutoff parameter in the PMI setting drastically changes the results. It is difficult to always know what settings were used in each attempt at this task, but the normalized experiments in this paper illustrate that the new cloze experiments have a heavy bias to the high-frequency events, regardless of how the events themselves are formalized (e.g., single argument or multi argument).

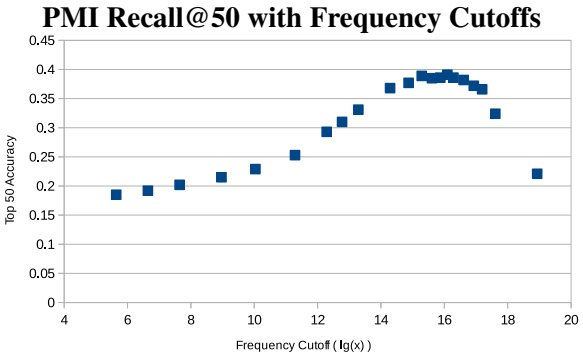

Figure 1: Frequency cutoffs. Events seen less than the cutoff are not included in the PMI ranking.

Unigrams	Bigrams	PMI
5.8	6.7	**9.4**

Table 3: Avg. information content of predictions.

7 Conclusion and Recommendations

Automatically generating event chains for evaluation does not test *relevant* script knowledge. The information content scores illustrate the huge extent to which common events (said, tell, went) dominate. More concerning, we can simply adjust the frequency cutoff in PMI learning, and it eliminates "poor results" from multiple previous papers. Language modeling approaches tend to capture frequent event patterns, not script knowledge.

Cloze Test	Unigram	Bigram	PMI
X scored	X said	X made	X scored
X set up	X have	X said	X beat
X headed	X had	X scored	X played
X challenged	told X	X accused	X missed
------??------	X told	X had	X hit
	X is	told X	X led
	X was	X told	X joined
	said X	X is	X went
	X has	X was	X finished
	killed X	said X	X opened

Table 4: Example Cloze test and the top predictions from ngrams and PMI

This is revealed in our frequency-based results, as well as in subjective error analysis like Table 4.

The core problem is that auto-generation does not evaluate script knowledge. We can't include all coreference chains from all documents and hope that this somehow measures script knowledge. The contradictory results from frequent events is just a symptom of the larger problem. We believe a *human annotator* should be in the loop for a meaningful evaluation. The test should include *meaningful core events*, and avoid others that are not script-relevant, such as discourse-related events (e.g., reporting verbs). Further, the test must not include events brought in through parser and coreference errors. By evaluating on parser output as gold data, we evaluate how well our models match our flawed text pre-processing tools. We acknowledge that human involvement is expensive, but the current trend to automate evaluations does not appear to be evaluating commonsense knowledge.

Finally, although this paper focuses on the narrative event cloze, we recognize that different evaluations are also possible. However, the traits of *human-annotation* and *core-events* seem to be required. One interesting task this year is Mostafazadeh et al. (2016) and the *Story Cloze* (manually created). Different from event chains, it still meets the requirements and provides a very large common corpus with 100k short stories. Another recent proposal is the InScript Corpus from Modi et al. (2016). They used Amazon Turk to create 1000 stories covering 10 predefined scenarios. While not as large and diverse as the Story Cloze, the entire corpus was annotated for gold events, coreference, and entities. This is an interesting new resource that avoids many of the problems discussed above, although issues of an event's *coreness* to a narrative may still need to be addressed.

We ultimately hope this short paper helps clarify recent results, inspires future evaluation, and most of all encourages discussion.

Acknowledgments

This work was supported by a grant from the Office of Naval Research.

References

Simon Ahrendt and Vera Demberg. 2016. Improving event prediction by representing script participants. In *Proceedings of North American Chapter of the Association for Computational Linguistics*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the Association for Computational Linguistics (ACL)*, Hawaii, USA.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence*.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344. Association for Computational Linguistics.

Ashutosh Modi, Tatjana Anikina, Simon Ostermann, and Manfred Pinkal. 2016. Inscript: Narrative texts annotated with script information. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 16), Portoroz, Slovenia*, pages 3485–3493.

Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *Proceedings of the the SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of North American Chapter of the Association for Computational Linguistics*.

Karl Pichotta and Raymond J. Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, Gothenburg, Sweden, April.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, Phoenix, Arizona.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*.