Improving Trope Detection with Crowdsourced Examples

Stanford CS224N Custom Project

Jean-Péïc Chou
Department of Computer Science
Stanford University
jeanpeic@stanford.edu

Abstract

Tropes are cultural narrative conventions that shape our expectations of stories. They thoughtfully encompass our understanding of the world and the representation of our culture. The use and comprehension of tropes require advanced human language skills such as causal, motivational, or pragmatic inferences. Detecting them is still an open and challenging research question that might be key to build ‘intelligent’ machines. In this regard, we propose to leverage the knowledge gathered by thousands of passionate people about tropes on tvtropes.org. Our contribution is twofold. First, we provide a dataset of specific examples for each trope extracted from the tropes wiki. From this scraped data, we propose the original task of the classification of tropes’ examples. It offers the main advantages of being more workable and interpretable due to the concision and relevance of the examples annotated by the community of tropers. Second, we present attempts to tackle this new challenge based on a word and sentence-level analysis and show the potential applications of such models on the more general task of detecting tropes in larger texts such as synopses. The proposed models outperform the state-of-the-art network when trained on individual tropes by making use of different levels of cues (word, sentence, or full context) based on the nature of the trope studied. This confirms the need to adapt the detection method or to build a multi-level comprehension model.

1 Key Information to include

- Mentor: Anna Goldie
- External Collaborators (if you have any): None
- Sharing project: No

2 Introduction

Stories are the pillars of human cultures. They shape our imagination, forge our understanding of the world, and ensure our cultural heritage. As such, the exploration of stories has been fascinating many philosophers and scientists since the beginning of the 20th century. Russian formalist Vladimir Propp was one of the firsts to systematically attempt to break down tales in a few categories [1]. Many other derived models and substantial story grammars were later conceived to decompose stories. However, most of them have been discredited because of the finiteness of the elements in their lexicon [2], because of their high and intricate abstraction in computational contexts, or their lack of data and examples.

Overcoming these deficiencies, a whole vocabulary of narrative units called tropes has been imagined and inventoried by a community of thousands of enthusiasts on the website tvtropes.org. Tropes are
narrative devices or storytelling conventions. They are recognizable patterns found in all kinds of media. Tropes can describe every level of a wok: the story and its discourse, characters and their interactions, location, time… For instance, the *Save the Princess* trope depicts the universal story plot in which a character, often portrayed by the *Damsel in Distress* trope, is kidnapped, and later rescued by the *Hero*. Tropes are highly sophisticated as they can hold abstract concepts related to morals, behaviors, or motivations. In this regard, the study of tropes can give access to human representations of the world and implicit knowledge they carry.

In this paper, we will tackle the challenge of trope detection in movie synopses introduced by Chen-Hsi Chang et al. [3]. We will work on the dataset TiMoS, containing 5,623 movie synopses as well as the trope appearances in each of the movies for 95 hand-picked tropes. In their work, the authors note the extreme difficulty of this task, requiring advanced linguistic, cognitive, and social skills, even for humans. A system able to achieve good results on this challenge could better tackle real world tasks such as recommendation systems, chat-bot, or opinion mining. With this dataset, the authors proposed MulCom, a multi-level comprehension network conceived for this task. If MulCom’s performances are very limited as the authors admit, it opens many avenues for improvement. In this regard, we propose to exploit external examples scraped on the website tvtropes.org where media works were thoroughly annotated, and preprocessed them to obtain more than 560,000 examples for 89 tropes. We then introduce the new closely related task of trope identification based on examples. Adapting binary classifiers based on word-level features and trained on this task to TiMoS notably yields largely better results than MulCom. Eventually, we propose a derived model of MulCom directly applicable to TiMoS and some leads to harness examples in this task.

### 3 Related Work

Stories have largely been exploited to assess machine comprehension performances. More specifically, movie synopses were analyzed through the lens of tags created by Sudipta Kar et al. in order to classify or compare them, as well as analyze their flow of emotions [4]. However, these tags remain quite shallow compared to the rich and dense library of tropes. Another way to probe movie synopses was proposed by Tapaswi et al. who introduced MovieQA [5], a dataset of plot synopses and multiple-choice questions aiming at evaluating story comprehension. Unfortunately, MovieQA’s possible answers were shown to be biased by B. Jasani et al. who demonstrated that the plot synopsis was not necessary to achieve satisfying results [6].

As of today, publications mentioning tropes are quite rare. In their study in trailer generation, Smith et al. were one of the firsts to cast light on tropes, describing weak tropes as visual elements characterizing genres [7]. Tropes have mostly been used as fuel for story generation [8] or social analyses [9]. They have only recently been considered as a vocabulary for advanced story grammars. In this regard, the website tvtropes was thoroughly examined [10][11]. Its structure was shown to provide a thoughtful backbone for such a grammar as well as rich information about tropes interactions. Trope detection was mentioned a few times but was tackled first by Chen-Hsi Chang et al. [3]. They conceived a Multi-Level Comprehension Network (MulCom) backed by a Multi-Step Recurrent Relational Network (MSRRN). The model aims at combining word-level information from Word2Vec, sentence-level information from a BERT encoder coupled with a RNN, and relation-level information from their MSRRN, a GNN based on characters’ interactions. Although the network architecture was built to be interpreted quite naturally, the role of some layers and operations remains unclear in the learning process. Besides, the authors note that even human evaluators were unable to detect some of the tropes when relying on the synopses alone. With commonsense inferences, the evaluators increased their final score by 30%, showing that additional cultural facts about stories and tropes are crucial in this task.

### 4 Approach

**Trope Identification.** In the first part of our work, we collected examples for each of the studied tropes by analyzing the structure of the website. Based on this new dataset, we propose the new original task of identifying the trope with examples as input. In practice, this task can be handled in two different ways.
This can be considered as a multi-class classification problem (MC). For each example, the model needs to decide which trope it is an example. This is the most natural way of using the dataset. However, in practice, sentences can be cues for several tropes at the same time. For instance, the \textit{Would Hurt a Child} trope that characterizes someone who is so evil that they might hurt a defenseless person is often found with the \textit{Big Bad} trope which corresponds to the main villain in a plot. Therefore, we can divide this task into as many binary classification tasks as there are tropes studied. The problem can then be considered as a multi-label classification one.

The perks of this challenge over the more global trope detection in synopses are that it is more workable and interpretable. Indeed, examples from \textit{vtvtropes} are generally a few sentences long and straightforward whereas synopses generally have 50 to 100 sentences and require wild guesses based on commonsense to detect some of the tropes.

To tackle the MC and ML tasks, we use and compare 3 different models on the dataset:

- A Support Vector Machine (SVM) trained on TF-IDF features
- A neural network with two BiLSTM layers followed by one dense layer using pretrained word vectors from GloVe as input \[^{[12]}\]
- The pretrained encoder BERT \[^{[13]}\] with a classification head on top

Once these models are trained, we want to apply them to TiMoS whose texts' nature are different from the crowdsourced examples. Based on the models trained for the MC and ML task:

- SS (Sentence by Sentence): We predict the probabilities of each class (trope) for each sentence of the synopses. For each class, we choose the final predicted probability as the maximum one over all sentences.
- FT (Full Text): We predict the probabilities of each class on the whole synopses.

The best threshold of the probability for each class if found on the training set of TiMoS.

**Trope Detection.** In this second part, we propose a derived model from MulCom \[^{[3]}\] aiming at removing some of its layers and operations and at making it more straightforward. We will call it MulCom2.

The task is a multi-label classification problem. Given a synopsis, the model predicts the appearance of each trope in it. The output is a binary vector \(T = t_1, t_2, \ldots\) where \(t_i\) corresponds to the appearance of trope \(i\) and \(|T|\) is the number of tropes studied. Following the intuition of authors of MulCom, we want to combine different levels of cues in order to detect tropes. Indeed, some tropes depend on single sentences or even words such as \textit{Would Hurt a Child} whereas \textit{Big Bad} might need more hints about a character throughout the story. The model is divided into 3 similar main parts: \textit{Knowledge}, \textit{Comprehension}, \textit{Classification}.

\textit{Knowledge.} The model stores a learnable embedding matrix representing the embeddings of tropes \(E \in \mathbb{R}^{|T| \times d_T}\) where \(T\) is the trope set and \(d_T\) is the dimension of the embeddings. \(E\) is randomly initialized and is used to as queries to perform attention without needing prior knowledge.

\textit{Comprehension.} The model extracts information from the text with different level of comprehension in each stream. For each of them, it uses tropes knowledge as queries of a multiplicative attention mechanism to keep the most valuable information.

- At the word-level, keys and values are word vectors from the pretrained GloVe model.
- At the sentence-level, keys and values are sentence embedded by the encoder BERT.
- At the paragraph-level, keys and values are paragraphs with the maximum length embedded by the encoder BERT (512 tokens).

\textit{Classification.} In this last step, the output of each stream is concatenated for each trope to obtain the output matrix \(M \in \mathbb{R}^{|T| \times (d_{GloVe} + d_{BERT} + d_{BERT})}\). It is then sent to a linear classifier that will give the final prediction. We use Binary Cross Entropy as our loss function to train on:

\[
\frac{1}{B} \sum_{b=1}^{B} \left( \frac{1}{T} \sum_{t=1}^{T} -[y_{b,t} \cdot \log(\hat{y}_{b,t}) + (1 - y_{b,t}) \cdot \log(1 - \hat{y}_{b,t})]\right)
\]
where $B$ is the batch, $y_{b,t}$ and $\hat{y}_{b,t}$ are the ground truth and prediction of the model of trope $t$ in batch $b$.

5 Experiments

5.1 Data

From the wiki tvtropes, we extracted over 760,000 examples from films, character pages, books, anime, comic books, manga, and TV shows for 89 tropes. Each example (text) is associated with exactly one trope (class/label). The scraped dataset was split into train, validation, and test sets.

TiMoS analyzed 93 tropes whereas we were only able to extract examples for 89 tropes. This is due to the fact that some tropes have been removed, renamed, or subdivided into multiple tropes. The results of our work will thus not be exactly equivalent. We also note that the level of information provided by each scraped example is very inconsistent. Some don’t give enough context (e.g. “Big Bad: Voldemort”) or on the contrary too much (e.g. by giving hundreds of inextricable lines of dialogue). In addition, the level of abstraction of the examples varies a lot making the dataset hard to read from the model’s perspective (e.g. “Voldemort is the main antagonist”, versus why or how he is, versus the description of a specific scene that depicts him like that…). We curated examples that we deemed appropriate for the model to learn properly by culling examples containing less than a certain number of words or examples that were incomprehensible without their context. The word clouds in Figure 2 demonstrate the relevance of the remaining examples. The Would Hurt a Child trope word cloud is mainly composed by the words “children”, “killing”, “try” or “threaten” for instance. Naturally, the Alcoholic character trope is described with alcohol-related words such as “drunk”, “alcohol”, or “liquor”.

Figure 1: MulCom2 is designed after MulCom [3] to deal with different levels of cues in movie synopses with multiple streams.

Figure 2: Word clouds of the examples of Would Hurt a Child (left) and The Alcoholic (right)
We are left with about 560,000 examples with a median of 6,032 examples per trope. The minimum number of examples is 1,003 (Cluster-F Bomb trope) while the maximum is 25,165 (Foreshadowing trope). Classes are therefore very imbalanced. For MC and ML task, we balance them by randomly sampling each class based on the minimum number of tropes.

For the individual binary classification tasks (ML), we also randomly sample the same number of tropes than the trope studied for each of them. In addition, we create for each trope a hard dataset (HD) that only comprises examples of tropes which are close to the trope studied. The closeness of two tropes is assessed based on the similarities of their word frequencies. Figure 3 shows a t-SNE visualization of tropes relying on the GloVe embedding of their 100 most frequent words. We can observe that Bittersweet Ending and Earn Your Happy Ending are close as they both relate to the end of a story (in blue on the left). Same thing goes for Big Bad and The Dragon (the Big Bad’s right hand) for instance (in read toward the middle).

![Figure 3: t-SNE visualization of tropes based on the embedding of their frequent words.](image)

5.2 Evaluation method

**Trope Identification.** The performances on the MC and ML tasks are evaluated with F1 and Area Under the ROC Curve (AUC) scores to determine how well the models can distinguish the classes no matter the threshold.

Models trained on MC and ML tasks are mainly compared to the best and worst performance of MulCom on individual tropes, i.e. respectively the Chekhov’s Gun trope and the Would Hurt a Child trope.

**Trope Detection.** On TiMoS test set, the performance is based on the F1 and mean average precision (mAP) metrics used by Chen-His Chang et al. [3] in order to compare our models with MulCom and the other baselines provided by the authors (Random, BERT [13] fine-tuned and Folksonomication [4]). In addition, We compare the models with the strategy of always detecting every trope as another baseline.
5.3 Experimental details

For MC and ML tasks, SVM and BiLSTM classifiers would take seconds to minutes to train and predict. For the first model, we tried several classifiers on tf-idf features (xgboost, random forest, MLP...) but SVM is the one that gave us the best result. For the second model, we tested different sizes of hidden layers and finally kept 64 for all of them. We also tried various pretrained word vectors such as Google’s Word2Vec or tried training it ourselves, but GloVe was the best model. We used Hugging face implementation of BERT pretrained model on uncased vocabulary for the last model. For each individual trope classifier, finetuning the full model would take 1 to 2 hours based on the evolution of the validation loss. For the MC version, the training runtime was around 12 hours as the dataset was much larger. We used batches of 16 examples over 3 epochs and stopped the training early if loss was not going in the right direction when evaluated. The learning rate was fixed to $2 \times 10^{-5}$.

The adaptation of the MC and ML models to TiMoS could be achieved in a lot of different ways. Instead of keeping the maximum probability predicted over all sentences, we tried using the whole average or the average of the N best (which would make sense for certain tropes). We tried also working by batch of sentences and by only considering some part of the synopses. The selected methods are the ones that gave the best results.

MulCom2’s trope embedding size was determined after a few tries and fixed at 64. In order to leverage the additional examples with this model, the goal was to use previously fine-tuned BERT model for the MC task or train it on TiMoS as well as the crowdsourced examples. Unfortunately, we were short on time as each training would take up to 8 hours. The slowest part of MulCom2’s process was the encoding of words by BERT which we would do externally (as BERT is only used as a feature extractor and not fine-tuned). We would then directly train the network by feeding it the vector representation of the synopses rather than the texts. Lacking time, MulCom2 was only trained and tested on synopses from TiMoS which took over 11 hours. It was supposed to be pretrained and trained on the examples as well as on synopses augmented by trope examples which would have required a day of computing.

Eventually, the process of scraping data from all media on tvtropes took approximately 15-20 hours.

5.4 Results

Results of the classification of the Woud Hurt a Child trope on the balanced dataset of examples and the hard one are shown in Table 1. We observe that BERT model is the one that performs the best with a F1 score of 92.41 and AUC of against 87.33 for SVM and 85.27 for the BiLSTM network. However, when applied to TiMoS to detect the same trope, the SVM model surprisingly outperforms all the other models including state-of-the-art MulCom with a F1 score of 21.67 when applied to the full text against 12.36 for MulCom. For both BERT and SVM, predicting on the full text returns better results than sentence by sentence which is not intuitive based on the nature of the trope. Indeed, a sentence describing a child being hurt or threatened should be a sufficient cue. Otherwise, SVM on hard examples struggles a bit more than on the randomly balanced dataset but this method does not make it more robust as its performance on TiMoS drops a bit.

<table>
<thead>
<tr>
<th>Examples</th>
<th>SVM</th>
<th>BiLSTM</th>
<th>BERT</th>
<th>SVM HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>87.33</td>
<td>85.27</td>
<td><strong>89.41</strong></td>
<td>83.21</td>
</tr>
<tr>
<td>AUC</td>
<td>93.42</td>
<td>92.12</td>
<td><strong>97.7</strong></td>
<td>91.36</td>
</tr>
<tr>
<td>TiMoS</td>
<td>SVM SS</td>
<td>SVM FT</td>
<td>BiLSTM SS</td>
<td>BERT SS</td>
</tr>
<tr>
<td>F1</td>
<td>16.91</td>
<td><strong>21.67</strong></td>
<td>15.62</td>
<td>16.9</td>
</tr>
</tbody>
</table>

Table 1: F1 and AUC Scores on the models trained on the dataset of examples for Would Hurt a Child (balanced and hard dataset) and F1 score on TiMoS

The results of the classification of the Chekhov’s Gun trope on the balanced and hard dataset shown in Table 2 are similar with BERT outperforming the other models on the binary classification task for both metrics. Here, BERT gives the best results with a F1 score of 40.61 against 38.58 for MulCom. This time, predicting the trope appearance sentence by sentence gives the best result which
is surprising as well as Chekhov’s Gun trope depicts “a plot device that is not significant until later in the story”. A broader grasp of the context would therefore be needed to detect it. Finally, we observe the same behavior for the model trained on the hard dataset.

### Table 2: F1 and AUC Scores on the models trained on the dataset of examples for Chekhov’s Gun (balanced and hard datasets) and F1 score on TiMoS

<table>
<thead>
<tr>
<th>Examples</th>
<th>SVM</th>
<th>BiLSTM</th>
<th>BERT</th>
<th>SVM HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>84.46</td>
<td>74.62</td>
<td>89.2</td>
<td>74.81</td>
</tr>
<tr>
<td>AUC</td>
<td>92.15</td>
<td>82.2</td>
<td>94.36</td>
<td>83.18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TiMoS</th>
<th>SVM SS</th>
<th>SVM FT</th>
<th>BiLSTM SS</th>
<th>BERT SS</th>
<th>BERT FT</th>
<th>SVM HD SS</th>
<th>SVM HD FT</th>
<th>MulCom</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>39.93</td>
<td>39.35</td>
<td>37.95</td>
<td>40.61</td>
<td>37.53</td>
<td>39.16</td>
<td>38.36</td>
<td>38.58</td>
</tr>
</tbody>
</table>

As for the results on the full list of tropes based on synopses shown in Table 3, BERT and SVM multi-class classifiers trained on examples are achieving correct results which are smaller but close to other baselines BERT fine-tuned and Folksonomication. We note here that our 3 proposed models are largely dominated by MulCom, especially for MulCom2 with a F1 score of 18.57 compared to 25.00 for its original counterpart.

The F1 score for each trope given by the multi-class models BERT and SVM are shown in Appendix for more details. While some tropes are barely detected, others such as Impaled with Extreme Prejudice or Eye Scream show very good results.

### Table 3: F1 and mAP Scores on the models trained on the dataset of examples for Chekhov’s Gun (balanced and hard) and F1 score on TiMoS

<table>
<thead>
<tr>
<th>Baseline</th>
<th>F1</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>13.97</td>
<td>8.14</td>
</tr>
<tr>
<td>Always Detect</td>
<td>14.75</td>
<td>8.09</td>
</tr>
<tr>
<td>BERT [13] (fine-tuned)</td>
<td>23.97</td>
<td>17.26</td>
</tr>
<tr>
<td>MulCom [3]</td>
<td><strong>25.00</strong></td>
<td><strong>18.73</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Our models</th>
<th>F1</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (trained on examples)</td>
<td>21.02</td>
<td>14.69</td>
</tr>
<tr>
<td>BERT (trained on examples)</td>
<td>22.71</td>
<td>16.14</td>
</tr>
<tr>
<td>MulCom2 (trained on synopses)</td>
<td>18.57</td>
<td>13.3</td>
</tr>
</tbody>
</table>

6 Analysis

The results show that word-level features such as simple tf-idf based on relevant examples can be very valuable in the detection of tropes. We note that the models’ performance varies a lot based on the trope detected and that one model does not dominate all the others. This makes sense as tropes describe different abstraction levels of a work. The detection of certain tropes significantly depends on the appearance of specific words such as Would Hurt a Child or The Alcoholic, while others require the full context such as Big Bad or the Chekhov’s Gun. Chen-Hsi et al. already noted that trope detection will need knowledge adaptation [3] depending on the trope by using different models. These results are overall a strong confirmation of this assessment.

The best results found by predicting sentence by sentence or on the full text for Would Hurt a Child and Chekhov’s Gun are surprising. For the first trope, the reasons could be that the synopses are not precise enough and the model needs to find cues from different parts of the full text, or that the model is easily tricked by specific words in some sentences as it only relies on tf-idf features. As for Chekhov’s Gun, if the plot device works at a high level, cues might be very specific and be found in certain sentences.
Some tropes such as *Irony* or *Hypocritical Humor* demand a deeper understanding of languages that might be too challenging for multi-class models. For all these quite abstract tropes or many other specific tropes, none of our models perform well, giving F1-scores rather low (down to 9.25 and 10.15 for *Exact Words*) compared to the worst F1-scores of MulCom (12.36) as observed in Figure 5 in Appendix. This important disparity of results explains why the overall F1 and mAP scores are lower than MulCom’s one. Overall, it should be remembered that these multi-class models were only trained on external examples having therefore no knowledge about the actual structure of synopses and their cues to certain tropes which is remarkable.

Finally, MulCom2 did not yield the expected results which is explainable by the fact that it is very rapidly overfitting due to its relatively high number of parameters compared to the number of synopses it was trained on as we can see if Figure 4. As it might need more data, the use of the additional examples for pretraining and training as it was planned might give better results. More importantly, MulCom2 does not extract character interaction features (out of the scope of this study) which the original MulCom heavily relies on according to its results [3].

Figure 4: The validation loss of MulCom2 keeps increasing after only 1 epoch while the model keeps overfitting the training set.

7 Conclusion

In this project, we have collected and preprocessed a dataset of more than 560,000 precise examples for 89 tropes that we make available for future research on tropes. We showed that this dataset could be leveraged to achieve satisfying results on the trope detection in movie synopses challenge by training classifiers on the original task of trope identification alone. In addition, we demonstrated the ability of these models to beat the state-of-the-art model when trained as binary classifier on single tropes rather than as multi-label classifiers. Eventually, we proposed a derived model of the state-of-the-art multi-level comprehension network which is a simpler and more straightforward model that still needs some adjustments.

We believe this work opens many avenues of research on tropes. This project is far from assessing the full potential of these examples in the task of trope detection. One lead could be to analyze these examples in their context by probing all the other trope examples, synopses, and metadata from the original work they come from. Furthermore, classifiers trained on trope identification are more interpretable than the other proposed networks for trope detection as they enable to break down synopses and give the location of the cues it found in the text. Such findings could be a crucial property to understand how tropes are sequenced, how they interact, or in which part of the stories they mostly intervene, which could lead to the foundations of a tropes grammar. Among many other applications, this new rich dataset could also be used to create stories by asking a decoder model to generate text based on tropes chosen by a user. Eventually, these leads might also be combined with the tropes knowledge graph extracted from *tvtropes*’ structure [11].
References


Figure 5: Results of SVM and BERT mutli-class models