Data-Driven Humor Recognition: Exploiting Incongruity-Based Features and Deep Learning Methods

Junze Li, Yubo Xie, and Pearl Pu
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
{junze.li,yubo.xie,pearl.pu}@epfl.ch

Abstract

Humor recognition has been widely studied as a text classification problem using data-driven approaches. However, most existing work does not examine the actual joke mechanism to understand humor. We break down any joke into two distinct components: the set-up and the punchline, and further explore the special relationship between them. Inspired by the incongruity theory of humor, we model the set-up as the part developing semantic uncertainty, and the punchline disrupting audience expectations. With increasingly powerful language models, we were able to feed the set-up along with the punchline into the GPT-2 language model, and calculate the uncertainty and surprisal values of the jokes. By conducting experiments on the SemEval 2021 Task 7 dataset, we found that these two features have better capabilities of telling jokes from non-jokes, compared with existing baselines. We also took part in the SemEval 2021 Task 7 (HaHackathon: Detecting and Rating Humor and Offense) by exploring and comparing different deep learning structures to recognize humor and offensiveness. Our DeBERTa model ranks top 3 in every subtask on the development phase leaderboard.

1 Introduction

Humor, regardless of age, gender, or cultural background, is perhaps one of the most fascinating human behaviors. Besides being able to provide entertainment, humor can also be beneficial to mental health by serving as a moderator of life stress (Lefcourt and Martin, 2012), and plays an important role in regulating human-human interaction. As Reeves and Nass (1996) have pointed out, people respond to computers in the same way as they do to real people, which indicates that modeling humor computationally could bring positive effects in human-computer interaction (Nijholt et al., 2003).

One of the important aspects of computational humor is to develop computer programs capable of recognizing humor in text. Early work on humor recognition (Mihalcea and Strapparava, 2005) proposed heuristic-based humor-specific stylistic features, for example alliteration, antonymy, and adult slang. More recent work (Yang et al., 2015; Chen and Soo, 2018; Weller and Seppi, 2019) regarded the problem as a text classification task, and adopted statistical machine learning methods and neural networks to train models on humor datasets. However, only few of the deep learning methods have tried to establish a connection between humor recognition and humor theories. Thus, one research direction in humor recognition is to bridge the disciplines of linguistics and artificial intelligence.

In this paper, we restrict the subject of investigation to jokes, one of the most common humor types in text form. As shown in Figure 1, these jokes usually consist of a *set-up* and a *punchline*. The set-up creates a situation that introduces the hearer into the story framework, and the punchline concludes the joke in a succinct way, intended to make the hearer laugh. Perhaps the most suitable humor theory for explaining such humor phenomenon is the *incongruity theory*, which states that the cause of laughter is the perception of something incongruous (the punchline) that violates the hearer's expectation (the set-up).

Based on the incongruity theory, we propose two features for humor recognition, by calculating the degree of incongruity between the set-up and the punchline. Recently popular pre-trained language models enable us to study such relationship based on large-scale corpora. Specifically, we fed the set-up along with the punchline into the GPT-2 language model (Radford et al., 2019), and obtained the surprisal and uncertainty values of the joke, indicating how surprising it is for the model to generate the punchline, and the uncertainty while generating it. We conducted experiments on a man-

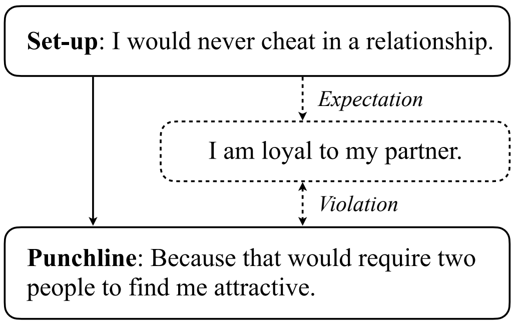

Set-up: I would never cheat in a relationship.
Expectation
I am loyal to my partner.
Violation
Punchline: Because that would require two people to find me attractive.

Figure 1: A joke example consisting of a set-up and a punchline. A violation can be observed between the punchline and the expectation.

Text	Humor Label	HR	HC	OR
...	1	1.126	0	3.098
...	0			1.282
...	1	3.983	1	1.644

Table 1: SemEval 2021 Task 7 dataset example. HR means humor rating score, HC means humor controversy label and OR means offense rating score.

ually labeled humor dataset, and the results showed that these two features could better distinguish jokes from non-jokes, compared with existing baselines. Our work made an attempt to bridge humor theories and humor recognition by applying large-scale pre-trained language models, and we hope it could inspire future research in computational humor.

Besides proposing incongruity-based features, we also explored some deep learning structures and participated in the SemEval 2021 Task 7 (HaHackathon: Detecting and Rating Humor and Offense) [1]. The organizers collected labels and ratings of all texts from a balanced set of age groups from 18-70. The invited annotators also represent a variety of genders, political stances and income levels. There are three questions for each annotator: 1) Is the intention of this text to be humorous? (0 or 1) 2) [If it is intended to be humorous] How humorous do you find it? (1-5). 3) [If it is intended to be offensive] How offensive do you find it? (1-5). They also represented the subjectivity of humor appreciation with a controversy score. This examines the variance in humor ratings for each text. If the variance of a text was higher than the median variance of all texts, the humor of the text is labelled as controversial. Table 1 gives some samples of the dataset.

[1]https://semeval.github.io/SemEval2021/

2 Literature Review

Humor Theory The attempts to explain humor date back to the age of ancient Greece, where philosophers like Plato and Aristotle regarded the enjoyment of comedy as a form of scorn, and held critical opinions towards laughter. These philosophical comments on humor, also followed by early Christian thinkers, were summarized as the *superiority theory*, which states that laughter expresses a feeling of superiority over other people's misfortunes or shortcomings. Starting from the 18th century, two other humor theories began to challenge the dominance of the superiority theory: the *relief theory* and the *incongruity theory*. The relief theory argues that laughter serves to facilitate the relief of pressure for the nervous system. This explains why laughter is caused when people recognize taboo subjects—one typical example is the wide usage of sexual terms in jokes. The incongruity theory, supported by Kant (1790), Schopenhauer (1883), and many later philosophers and psychologists, states that laughter comes from the perception of something incongruous that violates the expectations. This view of humor fits well the types of jokes commonly found in stand-up comedies, where the set-up establishes an expectation, and then the punchline violates it. Morreall (2020) gives a more comprehensive examination of these traditional humor theories.

As an expansion of the incongruity theory, Raskin (1979) proposed the Semantic Script-based Theory of Humor (SSTH) by applying the semantic script theory. It posits that, in order to produce verbal humor, two requirements should be fulfilled: (1) The text is compatible with two different scripts; (2) The two scripts with which the text is compatible are opposite. The General Theory of Verbal Humor (GTVH) (Attardo and Raskin, 1991) expanded the range of descriptive and explanatory dimensions of SSTH to six, called knowledge resources.

Humor Data Mihalcea and Strapparava (2005) created a one-liner dataset with humorous examples extracted from webpages with humor theme and non-humorous examples from Reuters titles, British National Corpus (BNC) sentences, and English Proverbs. Yang et al. (2015) scraped puns from the Pun of the Day website[2] and negative examples from various news websites. There is also work on the curation of non-English humor datasets (Zhang

[2]http://www.punoftheday.com/

Dataset	Size	# Turns
One-liner	32k	1
Pun of the Day	4.8k	1
Stupidstuff	3.77k	1
Wocka	10k	1
Short Jokes	232k	1
Reddit Jokes	195k	2
UR-FUNNY	16k	Multi
BigBang Theory	40k	Multi

Table 2: Comparison of different English humor datasets. # Turns denotes the number of turns of each sample. When a sample consists of a set-up and a punchline, # Turns is 2. When a sample is extracted from a dialogue, # Turns is multi.

et al., 2019; Blinov et al., 2019). Hasan et al. (2019) developed UR-FUNNY, a multimodal humor dataset that involves text, audio and video information extracted from TED talks. Bertero and Fung (2016) constructed the BigBang Theory dataset which contains dialogues annotated with punchlines. Some other datasets are not proposed in publications but released on Github, like Reddit Jokes, Stupidstuff, Wocka [3] and Short Jokes [4]. Table 2 compares different English humor datasets in detail.

Humor Recognition Most of the existing work on humor recognition in text focuses on one-liners, one type of jokes that delivers the laughter in a single line. The methodologies typically fall into two categories: feature engineering and deep learning. Mihalcea and Strapparava (2005) designed three human-centric features (alliteration, antonymy and synonym) for recognizing humor in the curated one-liner dataset. Mihalcea et al. (2010) approached the problem by calculating the semantic relatedness between the set-up and the punchline (they evaluated 150 one-liners by manually splitting them into "set-up" and "punchline"). Morales and Zhai (2017) proposed a probabilistic model and leveraged background text sources (such as Wikipedia) to identify humorous Yelp reviews. Liu et al. (2018) proposed to model sentiment association between elementary discourse units and designed features based on discourse relations. With neural networks being popular in recent years, some deep learning structures have been developed for the recognition of humor in text. Chen and Lee (2017) and Chen and Soo (2018) adopted convolutional neural networks, while Weller and Seppi (2019) used a Transformer architecture to do the classification task. Fan et al. (2020a,b) incorporated extra phonetic and semantic (ambiguity) information into the deep learning framework.

Although the work of Mihalcea et al. (2010) is the closest to ours, we are the first to bridge the incongruity theory of humor and large-scale pre-trained language models. Other work (Bertero and Fung, 2016) has attempted to predict punchlines in conversations extracted from TV series, but their subject of investigation should be inherently different from ours—punchlines in conversations largely depend on the preceding utterances, while jokes are much more succinct and self-contained.

3 Construct the Reddit Humor Dataset

In order to do data-driven humor recognition tasks, we should construct a humor dataset which contains balanced positive and negative samples. In Table 2, we can see that the Reddit Jokes dataset contains 2 turns sample which is suitable for our approach. So we firstly clean the Reddit Jokes dataset and extract the corresponding negative samples on Reddit website [5].

3.1 Data Cleaning

We cleaned the Reddit Jokes dataset with the following rules: 1) Remove jokes that are too long (>30 tokens, contain particular escape characters). 2) Remove jokes that have a set-up with fewer than 4 tokens. 3) Remove URLs in jokes (usually with pattern '[...](url)'). 4) Remove jokes whose punchline and set-up overlap (usually in this case, the set-up is literally a title, and the punchline is the full joke). For unigram, bigram, 3-gram and 4-gram, calculate the percentage of n-grams in the set-up that also appear in the punchline. If this percentage is higher than 0.5, then reject this joke. After cleaning procedure, we reduced the Reddit jokes from 194,553 to 119,235, which are the positive samples in Reddit Humor dataset.

3.2 Extract Negative Samples

The samples in Reddit Jokes dataset are scrapted from /r/jokes page of Reddit website. So we extracted samples from other pages as negative samples and implemented the similar data cleaning

[3] https://github.com/taivop/joke-dataset
[4] https://github.com/amoudgl/short-jokes-dataset
[5] https://www.reddit.com/

rules described in Section 3.1. To keep a balanced dataset, we selected 120,000 samples from all negative samples we extracted randomly. Finally, our Reddit Humor dataset contains 239,235 samples (119,235 positive ones and 120,000 negative ones).

4 Baseline Human-Centric Features

In order to recognize humor in natural language, we tried effective features in three aspects: (a) Phonetic Style; (b) Ambiguity Theory and (c) Semantic Relatedness. For each aspect, several features are designed to capture humor.

4.1 Phonetic Style

Ruch (2002) found that structural and phonetic properties of jokes are at least as important as their content regarding humor appreciation. Actually, one-liners often rely on the reader's awareness of attention-catching sounds, through linguistic phenomena such as alliteration, word repetition and rhyme, which produce a kind of comic effect even if the jokes are not necessarily meant to be humorous or not. Some similar rhetorical devices also play an important role in wordplay jokes, and are often used in newspaper headlines and in advertisement. The following one-liners are examples of jokes that include one or more alliteration chains:

Veni, Vidi, Visa: I came, I saw, I did a little shopping.

Infants don't enjoy infancy like adults do adultery.

An alliteration chain refers to two or more words beginning with the same phones. A rhyme chain is defined as the relationship that words end with the same syllable. We implemented the CMU Pronunciation Dictionary[6] to extract these phonetic features and computed the total number of chains found in the input text.

4.2 Ambiguity Theory

Semantic ambiguity is found to be a crucial part of humor jokes (Miller and Gurevych, 2015). Humor and ambiguity often come together when a listener expects one meaning, but the actual meaning is forced to be another one. Ambiguity occurs when the words in one sentence structure can be grouped in more than one way, thus yielding more than one understanding of readers, as shown in the example below.

Did you hear about the guy whose whole left side was cut off? He's all right now.

[6]http://www.speech.cs.cmu.edu/cgi-bin/cmudict

To capture the ambiguity included in a sentence, we took advantage of WordNet (Miller, 1995). We considered the possible meanings of each word and computed the ambiguity value:

$$\log \prod_{w \in s} \text{num\_of\_senses}(w), \tag{1}$$

where w is a word in the input text s.

4.3 Semantic Relatedness

Compared to the features constructed for one-liners, some methods try to capture the incongruity between the set-up and the punchline. The identification of incongruity in humor has to satisfy two requirements: 1) Jokes have to be coherent, which means the requirement for coherence between the set-up and the punchline. 2) They have to produce a surprising effect, which means the requirement of an unexpected punchline interpretation based on the set-up. In our settings, we assumed that jokes already satisfy the first requirement. So we emphasized the second requirement and tried to find models able to identify the surprising effect generated by the punchline.

We used several knowledge-based metrics to measure the semantic relatedness between the set-up and the punchline. The idea is that the real punchline will have a minimum semantic relatedness with respect to the set-up.

Before computing the semantic relatedness, we should disambiguity each word to determine the actual meaning in a given sentence. The Lesk algorithm is a classical algorithm for word sense disambiguation (Lesk, 1986). The Lesk algorithm is based on the assumption that words in a given context will tend to share a common topic. A simplified version of the Lesk algorithm is to compare the dictionary definition of an ambiguous word in WordNet with the terms contained in its context. The implementation looks like this: 1) For every sense of the ambiguous word should count the amount of words that are in the context of that word and in the dictionary definition of that sense. 2) The sense that is to be chosen is the sense which has the biggest number of this count.

Given a metric for word-to-word relatedness, we defined the semantic relatedness of two text segments T_1 and T_2 using a metric that combines the semantic relatedness of each text segment in turn with respect to the other text segment (Mihalcea et al., 2006): 1) For each word w in the segment T_1, we tried to identify the word in the segment T_2 that has the highest semantic relatedness, according to one of the word-to-word measures described below. 2) The same process is applied to determine the most similar word in T_1 starting with words in T_2. 3) The word similarities are weighted, summed up, and normalized with the length of each text segment. 4) The resulting relatedness scores are combined using a simple average.

We present below three word-to-word relatedness metrics found to work well based on WordNet. The first two metrics have the best performance in the work of Mihalcea et al. (2010) and the last one is based on shortest paths in WordNet.

- **Leacock & Chodorow similarity** (Leacock and Chodorow, 1998), defined as

$$\mathrm{Sim}_{lch} = -\log \frac{length}{2 * D}, \tag{2}$$

  where *length* is the length of the shortest path between two concepts using node-counting, and D is the maximum depth of WordNet.

- **Wu & Palmer similarity** (Wu and Palmer, 1994) calculates similarity by considering the depths of the two synsets in WordNet, along with the depth of their *LCS* (Least Common Subsumer), which is defined as

$$\mathrm{Sim}_{wup} = \frac{2 * \mathrm{depth}(LCS)}{\mathrm{depth}(C_1) + \mathrm{depth}(C_2)}, \tag{3}$$

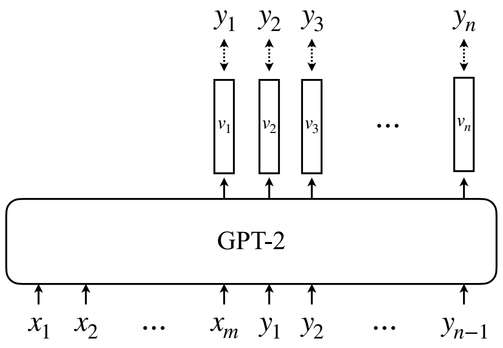
  where C_1 and C_2 denote synset 1 and synset 2 respectively.

- **Path similarity** (Rada et al., 1989) is also based on the length of the shortest path between two concepts in WordNet, which is defined as

$$\mathrm{Sim}_{path} = \frac{1}{1 + length}. \tag{4}$$

5 Uncertainty and Surprisal

As introduced in 4.3, the incongruity theory attributes humor to the violation of expectation. This means the punchline delivers the incongruity that turns over the expectation established by the set-up, making it possible to interpret the set-up in a completely different way. However, different from the traditional semantic metrics, pre-trained language models make it possible to study such relationship between the set-up and the punchline based on large-scale corpora, with neural networks blooming in recent years. Given the set-up, language models are capable of writing expected continuations, enabling us to measure the degree of incongruity, by comparing the actual punchline with what the language model is likely to generate.



y_1 y_2 y_3 y_n
v_1 v_2 v_3 ⋯ v_n
GPT-2
x_1 x_2 ⋯ x_m y_1 y_2 ⋯ y_{n-1}

Figure 2: The set-up x and the punchline y are concatenated and fed into GPT-2 for predicting the next token. v_i's are probability distributions on the vocabulary.

In this paper, we leverage the GPT-2 language model (Radford et al., 2019), a Transformer-based architecture trained on the WebText dataset consisting of over 8 million documents for a total of 40 GB of text. WebText was curated without making any assumptions on the genres of the text; thus the resulting model is domain independent, and is shown to learn many NLP tasks without explicit supervision. We chose GPT-2 as our research tool because: (1) GPT-2 is already pre-trained on massive data and publicly available online, which spares us the training process; (2) it is domain independent, thus suitable for modeling various styles of English text. Our goal is to model the set-up and the punchline as a whole piece of text using GPT-2, and analyze the probability of generating the punchline given the set-up. In the following text, we denote the set-up as x, and the punchline as y. Basically, we are interested in two quantities regarding the probability distribution $p(y|x)$: uncertainty and surprisal, which are elaborated in the next two sections.

5.1 Uncertainty

The first question we are interested in is: given the set-up, how uncertain it is for the language model to continue? This question is related to SSTH, which states that, for a piece of text to be humorous, it should be compatible with two different scripts. To put it under the framework of set-up and punchline, this means the set-up could have multiple ways of interpretation, according to the following punchline.

Thus, one would expect a higher uncertainty value when the language model tries to continue the set-up and generate the punchline.

We propose to calculate the averaged entropy of the probability distributions at all token positions of the punchline, to represent the degree of uncertainty. As shown in Figure 2, the set-up x and the punchline y are concatenated and then fed into GPT-2 to predict the next token. While predicting the tokens of y, GPT-2 produces a probability distribution v_i over the vocabulary. The averaged entropy is then defined as

$$U(x, y) = -\frac{1}{|y|} \sum_{i=1}^{n} \sum_{w \in V} v_i^w \log v_i^w, \tag{5}$$

where V is the vocabulary.

5.2 Surprisal

The second question we would like to address is: how surprising it is when the language model actually generates the punchline? As the incongruity theory states, laughter is caused when something incongruous is observed and it violates the previously established expectation. Therefore, we expect the probability of the language model generating the actual punchline to be relatively low, which indicates the surprisal value should be high. Formally, the surprisal is defined as

$$\begin{aligned} S(x, y) &= -\frac{1}{|y|} \log p(y|x) \\ &= -\frac{1}{|y|} \sum_{i=1}^{n} \log v_i^{y_i}. \end{aligned} \tag{6}$$

6 Deep Learning Methods

We implemented different deep learning structures to find the most effective one for recognizing humor, which are CNN, Bi-LSTM, Transformer and DeBERTa.

6.1 CNN

Convolutional neural network (CNN) is designed to learn some particular local features of high dimensional data such as images or speech signals. Most of CNN structures are designed for Computer Vision tasks. But CNN also shows successes in several text classification tasks (Kim, 2014; Johnson and Zhang, 2015).

We chose the CNN architecture following the Chen and Soo (2018) for text classification and Figure 3 shows the structure details. In the embedding layer, each tokenized input sentence is converted to a matrix by utilization of the GloVe word embedding vectors (Pennington et al., 2014). Next, we experimented with different filter sizes ranging from 3 to 20 to adapt to the average sentence length. The window width of each filter is set to the embedding dimension. For each filter size, 100-200 filters are applied in the convolutional layer. Then we implemented a max pooling layer to flatten the output of convolutional layer. The output dimension of the max pooling layer equals to the number of filters we choose.

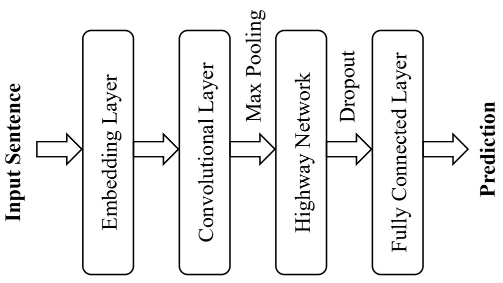


Figure 3: The CNN Architecture.

In our model, we also used the highway network layer after max pooling layer to improve the model's performance (Srivastava et al., 2015). The highway network involves shortcut connections with gate functions. These gates are data-dependent with parameters. It allows information unimpeded to flow through several layers like information highways. The architecture is characterized by the gate units that learn to regulate the flow of information through the network. With this architecture, we could train much deeper CNNs. In the end, the output from a fully connected layer is used to predict labels.

6.2 Bi-LSTM

Long Short Term Memory (LSTM) is proposed to overcome gradient exploding and long term dependency problems of Recurrent Neural Network (RNN) (Hochreiter and Schmidhuber, 1997). There are one cell and three gates in LSTM structure, which are a memory cell c_t, an input gate i_t, a forget gate f_t and an output gate o_t. The computations are indicated by the following equations,

$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + V_i c_{t-1}\right), \tag{7}$$
$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + V_f c_{t-1}\right), \tag{8}$$
$$o_t = \sigma\left(W_o x_t + U_o h_{t-1} + V_o c_{t-1}\right), \tag{9}$$

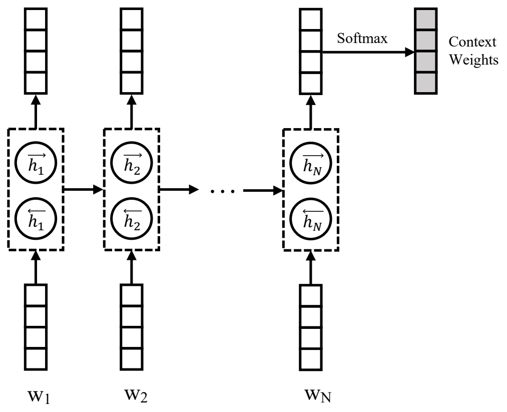


Figure 4: The Bi-LSTM Layer Structure.

$$\tilde{c_t} = \tanh\left(W_c x_t + U_c h_{t-1}\right), \quad (10)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad (11)$$
$$h_t = o_t \odot \tanh\left(c_t\right), \quad (12)$$

where x_t is the input vector at each time step t, h_t is the hidden vector at each time step t, $W_{i,f,o,c}$, $U_{i,f,o,c}$ and $V_{i,f,o,c}$ are learnable parameters. In LSTM structure, the hidden state h_t only encodes the input text in a forward direction without cosidering the backward direction.

Bidirectional LSTM (Bi-LSTM) is designed to improve LSTM structure which consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction (Graves and Schmidhuber, 2005). Bi-LSTM effectively increases the amount of information available to the network and improves the context available to the algorithm by knowing what words immediately follow and precede a word in a sentence. Figure 4 shows the Bi-LSTM structure, where a forward LSTM and a backward LSTM concatenate the two hidden state vectors as the representation of the current word. The equations are as following,

$$\overrightarrow{h}_t = H\left(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right), \quad (13)$$
$$\overleftarrow{h}_t = H\left(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}\right), \quad (14)$$
$$h_{\text{out}} = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y, \quad (15)$$

where $\overrightarrow{h}_t$ denotes the forward LSTM, $\overleftarrow{h}_t$ denotes the backward LSTM and h_{out} is the final output of the Bi-LSTM layer.

6.3 Transformer

Unlike recurrent neural networks that process text in sequence, Transformer applies self-attention to compute in parallel every word from the input text

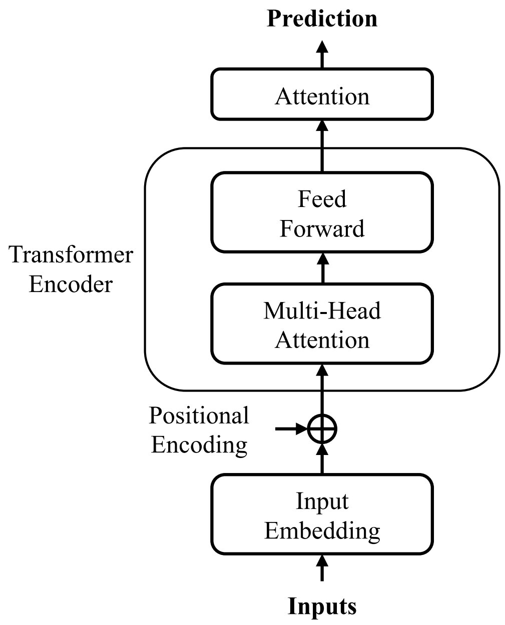


Figure 5: The Transformer Architecture.

an attention weight that represents the influence each word has on another (Vaswani et al., 2017). Transformer consists stacked encoder and decoder, following the auto-encoder structure. In text generation tasks, the encoder transforms an input sequence of words $X = (x_1, x_2, ..., x_n)$ into a sequence of continuous representations $z = (z_1, z_2, ..., z_n)$. Given z, the decoder then generates an output sequence of tokens $Y = (y_1, y_2, ..., y_m)$ one token at each time step. The decoder is auto-regressive at each time step, given the previous generated token as additional input when generating the next. The objective function can be written as

$$p(Y|X) = p(y_1|z) \prod_{t=2}^{m} p(y_t|z, y_1, ..., y_{t-1}). \quad (16)$$

However in our humor recognition task, we only need the encoder part to compute hidden vectors of each input sentence. Figure 5 shows the architecture of our Transformer model and some mechanisms involved in this architecture will be introduced.

6.3.1 Encoder Stack

The encoder stack consists of 6 identical layers. Each identical layer has one multi-head attention sub-layer and one feed forward sub-layer. There is also a residual connection around each of the two sub-layers, followed by layer normalization (Ba et al., 2016).

6.3.2 Self-Attention Mechanism

Each embedding vector is mapped to three vectors, query(q), key(k) and value(v) by three corresponding mapping matrices. The output is computed as a weighted sum of values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. In practice, we packed all the queries, keys and values vectors into Q, K and V matrices. We computed the self-attention α as

$$\alpha(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{17}$$

where d_k denotes the dimension of q. The multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions, which is computed as

$$\beta = \text{concat}(\alpha_1, ..., \alpha_h)W^O, \tag{18}$$

$$\alpha_i = \alpha(QW_i^Q, KW_i^K, VW_i^V), \tag{19}$$

where β denotes the multi-head attention, W^O, W_i^Q, W_i^K and W_i^V are all projection matrices, which are corresponding to the dimension of q, k and v. In multi-head attention mechanism, different α_i may focus on different parts of the input sentence in order to extract more features of the input sentence.

6.3.3 Feed-Forward Network

In addition to the multi-head attention layers, both encoder and decoder contain fully connected feed-forward networks. The function of this feed-forward network can be written as

$$\gamma(x) = \max(0, xW_1 + b_1)W_2 + b_2, \tag{20}$$

where γ denotes the output, W_1, W_2 are weight matrices and b_1, b_2 are biases. This layer consists of two linear transformations with a ReLU activation function.

6.3.4 Positional Encoding

Up to now, we do not make use of the sequence order of the input sentence. We implemented the positional encoding to extract the position information of the input sentence. The computation of the positional encoding is

$$C_{\text{pos},2i} = \sin(\text{pos}/10000^{2i/d_{\text{model}}}), \tag{21}$$

$$C_{\text{pos},2i+1} = \cos(\text{pos}/10000^{2i/d_{\text{model}}}), \tag{22}$$

where pos denotes the position and i denotes the dimension of encoding. d_{model} is the dimension of the input vector and output vector.

6.4 DeBERTa

Decoding-enhanced BERT with disentangled attention (DeBERTa) improves the Transformer-based models using two novel techniques, disentangled attention mechanism and enhanced mask decoder (He et al., 2020). In our humor recognition task, we only need the encoder part so we mainly introduce the disentangled attention mechanism here.

For a token at position i in a sequence, we represent it using two vectors, H_i and $P_{i|j}$, which represent its content and relative position with the token at position j, respectively. The calculation of the cross attention score between tokens i and j can be denoted as

$$\begin{aligned} A_{i,j} &= \{H_i, P_{i|j}\} \times \{H_j, P_{j|i}\}^\top \\ &= H_i H_j^\top + H_i P_{j|i}^\top + P_{i|j} H_j^\top + P_{i|j} P_{j|i}^\top. \end{aligned} \tag{23}$$

The attention weight of a word pair can be computed as a sum of four attention scores using disentangled matrices on their contents and positions as *content-to-content*, *content-to-position*, *position-to-content*, and *position-to-position*.

Shaw et al. (2018) used a separate embedding matrix to compute the relative position bias in computing attention weights. This is equivalent to computing the attention weights using only the *content-to-content* and *content-to-position* terms in Equation 23. However, the *position-to-content* term is also important since the attention weight of a word pair depends not only on their contents but on their relative positions, which can only be fully modeled using both the content-to-position and position-to-content terms.

In Section 6.3.2, we introduce what is the self-attention mechanism. Here we represent how disentagled attention mechanism works. Denote k as the maximum relative distance, $\delta(i,j) \in [0, 2k)$ as the relative distance from token i to token j, which is defined as

$$\delta(i,j) = \begin{cases} 0 \text{ for} & i - j \leqslant -k \\ 2k - 1 \text{ for} & i - j \geqslant k \\ i - j + k \text{ others} . \end{cases} \tag{24}$$

Then the disentangled attention can be computed as

$$Q_c = HW_{q,c}, K_c = HW_{k,c}, V_c = HW_{v,c}, \tag{25}$$

$$Q_r = PW_{q,r}, K_r = PW_{k,r}, \tag{26}$$

$$A_{i,j} = Q_i^c K_j^{c\top} + Q_i^c K_{\delta(i,j)}^{r\top} + K_j^c Q_{\delta(j,i)}^{r\top}, \tag{27}$$

$$\alpha = \text{softmax}(\frac{A}{\sqrt{3d}})V_c, \tag{28}$$

where H represents input hidden vectors, Q_c, K_c and V_c are projected content vectors generated using projection matrices $W_{q,c}$, $W_{k,c}$ and $W_{v,c}$ respectively, P represents the relative position embedding vectors, Q_r and K_r are projected relative position vectors generated using projection matrices $W_{q,r}$ and $W_{k,r}$ respectively, $A_{i,j}$ is the element of attention matrix A denoting the attention score from token i to token j, Q_i^c is the i-th row of Q_c, K_j^c is the j-th row of K_c, $K^r_{\delta(i,j)}$ is the $\delta(i,j)$-th row of K_r with regarding to relative distance $\delta(i,j)$ and $Q^r_{\delta(j,i)}$ is the $\delta(j,i)$-th row of Q_r with regarding to relative distance $\delta(j,i)$.

7 Experiments

7.1 Experiment with Features

We evaluated and compared the proposed features with several baselines by conducting experiments in two settings: predicting using individual features, and combining the features with a content-based text classifier.

7.1.1 Baselines

In Section 4, we introduce the baseline features constructed for humor recognition and understanding. We chose alliteration, ambiguity and semantic relatedness (computed by Leacock & Chodorow similarity, Wu & Palmer similarity and path similarity respectively) as our baseline features.

7.1.2 Dataset

The released training set in SemEval 2021 Task 7 contains 8,000 manually labeled examples in total, with 4,932 being positive, and 3,068 negative. To adapt the dataset for our purpose, we only considered positive examples with exactly two sentences, and negative examples with at least two sentences. For positive examples (jokes), the first sentence was treated as the set-up and the second the punchline. For negative examples (non-jokes), consecutive two sentences were treated as the set-up and the punchline, respectively.[7] After splitting, we cleaned the data with the following rules: (1) we restricted the length of set-ups and punchlines to be under 20 (by counting the number of tokens); (2) we only kept punchlines whose percentage of alphabetical letters is greater than or equal to 75%; (3) we discarded punchlines that do not begin with an alphabetical letter. As a result, we obtained 3,341 examples in total, consisting of 1,815 jokes and 1,526 non-jokes. To further balance the data, we randomly selected 1,526 jokes, and thus the final dataset contains 3,052 labeled examples in total. For the following experiments, we used 10-fold cross validation, and the averaged scores are reported.

[7]We refer to them as set-up and punchline for the sake of convenience, but since they are not jokes, the two sentences are not real set-up and punchline.

	P	R	F1	Acc
Random	0.4973	0.4973	0.4958	0.4959
Sim_{lch}	0.5291	0.5179	0.4680	0.5177
Sim_{wup}	0.5289	0.5217	0.4919	0.5190
Sim_{path}	0.5435	0.5298	0.4903	0.5291
Alliteration	0.5353	0.5349	0.5343	0.5354
Ambiguity	0.5461	0.5365	0.5127	0.5337
Uncertainty	0.5840	0.5738	0.5593	0.5741
Surprisal	0.5617	0.5565	0.5455	0.5570
U+S	**0.5953**	**0.5834**	**0.5695**	**0.5832**

Table 3: Performance of individual features. Last row (U+S) is the combination of uncertainty and surprisal. P: Precision, R: Recall, F1: F1-score, Acc: Accuracy. P, R, and F1 are macro-averaged, and the scores are reported on 10-fold cross validation.

7.1.3 Predicting Using Individual Features

To test the effectiveness of our features in distinguishing jokes from non-jokes, we built an SVM classifier for each individual feature (uncertainty and surprisal, plus the baselines). The resulted scores are reported in Table 3. Compared with the baselines, both of our features (uncertainty and surprisal) achieved higher scores for all the four metrics. In addition, we also tested the performance of uncertainty combined with surprisal (last row of the table), and the resulting classifier shows a further increase in the performance. This suggests that, by jointly considering uncertainty and surprisal of the set-up and the punchline, we are better at recognizing jokes.

7.1.4 Boosting a Content-Based Classifier

We also evaluated the proposed features as well as the baselines under the framework of a content-based classifier. The idea is to see if the features could further boost the performance of existing text classifiers. To create a starting point, we encoded each set-up and punchline into vector representations by aggregating the GloVe (Pennington et al., 2014) embeddings of the tokens (sum up and then normalize by the length). We used the GloVe embeddings with dimension 50, and then concatenated the set-up vector and the punchline vector, to represent the whole piece of text as a vector of dimension

	P	R	F1	Acc
GloVe	0.8233	0.8232	0.8229	0.8234
GloVe+Sim$_{lch}$	0.8255	0.8251	0.8247	0.8250
GloVe+Sim$_{wup}$	0.8264	0.8260	0.8254	0.8257
GloVe+Sim$_{path}$	0.8252	0.8244	0.8239	0.8244
GloVe+Alliter.	0.8299	0.8292	0.8291	0.8297
GloVe+Amb.	0.8211	0.8203	0.8198	0.8201
GloVe+U	0.8355	0.8359	0.8353	0.8359
GloVe+S	0.8331	0.8326	0.8321	0.8326
GloVe+U+S	**0.8368**	**0.8368**	**0.8363**	**0.8365**

Table 4: Performance of the features when combined with a content-based classifier. U denotes uncertainty and S denotes surprisal. P: Precision, R: Recall, F1: F1-score, Acc: Accuracy. P, R, and F1 are macro-averaged, and the scores are reported on 10-fold cross validation.

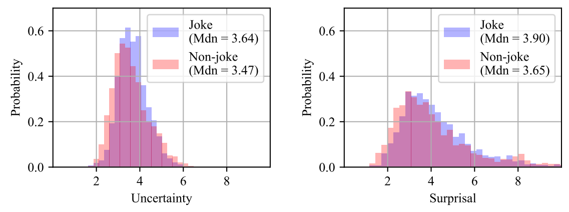


Figure 6: Histograms of uncertainty (left) and surprisal (right), plotted separately for jokes and non-jokes. Mdn stands for Median.

100. For each of the features (uncertainty and surprisal, plus the baselines), we appended it to the GloVe vector, and built an SVM classifier to do the prediction. Scores are reported in Table 4. As we can see, compared with the baselines, our features produce larger increases in the performance of the content-based classifier, and similar to what we have observed in Table 3, jointly considering uncertainty and surprisal gives further increase in the performance.

7.1.5 Visualizing Uncertainty and Surprisal

To get a straightforward vision of the uncertainty and surprisal values for jokes versus non-jokes, we plot their histograms in Figure 6 (for all 3,052 labeled examples). It can be observed that, for both uncertainty and surprisal, jokes tend to have higher values than non-jokes, which is consistent with our expectations in Section 5.

7.2 SemEval 2021 Task 7 Results

We implemented different deep learning structures introduced in Section 6 and compared model performance based on the metrics in each subtask to find the most effective model for humor recognition.

	P	R	F1	Acc
CNN	0.8016	0.8107	0.8053	0.8160
Bi-LSTM	0.8616	0.8400	0.8485	0.8630
Transformer	0.8949	0.9430	0.9183	0.8940
DeBERTa	**0.9707**	**0.9446**	**0.9575**	**0.9470**

Table 5: Performance of different deep learning models on SemEval 2021 Task 7 Subtask 1a (Humor Detection).

7.2.1 Model Settings

The max length of input sentences is 100 and the optimization algorithm is Adam for all models. In CNN and Bi-LSTM model, we initialized the embedding layer with 200 dimension GloVe word embeddings and the vocabulary size is 30,004. We chose 100 filters with kernel size 5 for CNN layer and 32 LSTM units for Bi-LSTM layer. In the Transformer model, there are 12 encoders and 12 multi-heads, hidden size is 768, vocabulary size is 50,265, learning rate is 5×10^{-5} and we took advantage of RoBERTa word embedding initialization. In the DeBERTa model, there are 24 encoders and 16 multi-heads, hidden size is 1024 and learning rate is 2×10^{-6}. The batch size is 32 in CNN, Bi-LSTM and Transformer model but 16 in DeBERTa model.

7.2.2 Dataset

We trained different models on the training dataset from SemEval 2021 Task 7 which contains 8,000 samples and evaluated the model performance on the development dataset used in the leaderboard evaluation which contains 1,000 samples.

7.2.3 Results

In the Subtask 1a (Humor Detection), the resulted scores are reported in Table 5. It can be observed that DeBERTa model outperforms other models a lot in both accuracy and F1-score, which indicates DeBERTa model has the ability to extract and learn humor properties more accurately and effectively. Therefore, we implemented DeBERTa model to complete other subtasks.

We represent the current top 5 leaderboard in the development phase and our model performance in Table 6. Our model ranks top 3 in all subtasks.

7.2.4 Discussion

Overall, it can be observed that Transformer-based models perform better than sequence model or CNN model. By comparing Transformer model and DeBERTa model, the disentangle attention mechanism in DeBERTa model extracts more informative re-

SemEval Task7	Subtask 1a		Subtask 1b	Subtask 1c		Subtask 2
Leaderboard	F1	Acc	RMSE	F1	Acc	RMSE
Rank 1	0.9603	0.9500	0.4848	0.6598	0.5301	0.4817
Rank 2	0.9570	0.9450	0.4853	0.6567	0.4905	0.4854
Rank 3	0.9539	0.9410	0.4936	0.6565	0.5000	0.5064
Rank 4	0.9538	0.9410	0.4952	0.6553	0.4873	0.5209
Rank 5	0.9528	0.9400	0.5004	0.6546	0.5158	0.5239
Our Team	0.9575	0.9470	0.4923	0.6595	0.5016	0.4794
Our Rank	**2/59**		**3/35**	**2/33**		**1/29**

Table 6: The top 5 leaderboard and our model performance in development phase.

lation in each word pair of the input sentence. In the Transformer structure, the position embedding vectors are only added to the input vectors of the self-attention layer. But in the DeBERTa structure, the relative position vectors are calculated in the self-attention layer by disentangled matrices. Since both the meaning and position of each word are crucial for humorous effect, the DeBERTa model can detect humor in a better way.

8 Conclusion

In this paper, we first made an attempt in establishing a connection between the humor theories and the nowadays popular pre-trained language models. Based on that, we proposed two features related to the incongruity theory of humor: uncertainty and surprisal. We conducted experiments on a humor dataset, and the results suggest that our approach has an advantage in humor recognition over the baselines. The proposed features can also provide insight for the task of two-line joke generation—when designing the text generation algorithm, one could exert extra constraints so that the set-up is chosen to be compatible with multiple possible interpretations, and the punchline should be surprising in a way that violates the most obvious interpretation. We hope our work could inspire future research in the community of computational humor.

In the process of designing the system involved in SemEval 2021 Task 7 Competition, we found that DeBERTa structure provides a promising research direction of understanding not only humor but other complex emotions contained in nature language. In the future work, we could add some extra embedding vectors representing phonetic style or ambiguity property of each word in the self-attention layer. The computation of relative position vectors and disentangled matrices could also be optimized in other NLP tasks.

References

Salvatore Attardo and Victor Raskin. 1991. Script theory revis(it)ed: Joke similarity and joke representation model. *Humor: International Journal of Humor Research*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dario Bertero and Pascale Fung. 2016. A long short-term memory framework for predicting humor in dialogues. In *Proceedings of NAACL-HLT 2016*, pages 130–135.

Vladislav Blinov, Valeria Bolotova-Baranova, and Pavel Braslavski. 2019. Large dataset and language model fun-tuning for humor recognition. In *Proceedings of ACL 2019*, pages 4027–4032.

Lei Chen and Chong Min Lee. 2017. Convolutional neural network for humor recognition. *CoRR*, abs/1702.02584.

Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. In *Proceedings of NAACL-HLT 2018, Volume 2 (Short Papers)*, pages 113–117.

Xiaochao Fan, Hongfei Lin, Liang Yang, Yufeng Diao, Chen Shen, Yonghe Chu, and Tongxuan Zhang. 2020a. Phonetics and ambiguity comprehension gated attention network for humor recognition. *Complex.*, 2020:2509018:1–2509018:9.

Xiaochao Fan, Hongfei Lin, Liang Yang, Yufeng Diao, Chen Shen, Yonghe Chu, and Yanbo Zou. 2020b. Humor detection via an internal and external neural network. *Neurocomputing*, 394:105–111.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.

Md. Kamrul Hasan, Wasifur Rahman, AmirAli Bagher Zadeh, Jianyuan Zhong, Md. Iftekhar Tanveer, Louis-Philippe Morency, and Mohammed (Ehsan) Hoque. 2019. UR-FUNNY: A multimodal language dataset for understanding humor. In *Proceedings of EMNLP-IJCNLP 2019*, pages 2046–2056.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *NAACL HLT 2015*, pages 103–112.

Immanuel Kant. 1790. Critique of judgment, ed. and trans. *WS Pluhar, Indianapolis: Hackett*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and WordNet sense similarity for word sense identification. In *WordNet, An Electronic Lexical Database*. The MIT Press.

Herbert M Lefcourt and Rod A Martin. 2012. *Humor and life stress: Antidote to adversity*. Springer Science & Business Media.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation, SIGDOC 1986, Toronto, Ontario, Canada, 1986*, pages 24–26. ACM.

Lizhen Liu, Donghai Zhang, and Wei Song. 2018. Modeling sentiment association in discourse for humor recognition. In *Proceedings of ACL 2018, Volume 2 (Short Papers)*, pages 586–591.

Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of HLT/EMNLP 2005*, pages 531–538.

Rada Mihalcea, Carlo Strapparava, and Stephen G. Pulman. 2010. Computational models for incongruity detection in humour. In *Proceedings of CICLing 2010*, volume 6008 of *Lecture Notes in Computer Science*, pages 364–374.

George A. Miller. 1995. Wordnet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of English puns. In *Proceedings of ACL 2015*, pages 719–729.

Alex Morales and Chengxiang Zhai. 2017. Identifying humor in reviews using background text sources. In *Proceedings of EMNLP 2017*, pages 492–501.

John Morreall. 2020. Philosophy of Humor. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, fall 2020 edition. Metaphysics Research Lab, Stanford University.

Anton Nijholt, Oliviero Stock, Alan J. Dix, and John Morkes. 2003. Humor modeling in the interface. In *CHI 2003 Extended Abstracts on Human Factors in Computing Systems*, pages 1050–1051.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP 2014*, pages 1532–1543.

Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. 1989. Development and application of a metric on semantic nets. *IEEE Trans. Syst. Man Cybern.*, 19(1):17–30.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Victor Raskin. 1979. Semantic mechanisms of humor. In *Annual Meeting of the Berkeley Linguistics Society*, volume 5, pages 325–335.

Byron Reeves and Clifford Nass. 1996. *The media equation: How people treat computers, television, and new media like real people and places*. Cambridge University Press.

Willibald Ruch. 2002. Computers with a personality? lessons to be learned from studies of the psychology of humor. In *Proceeding of The April Fools Day Workshop on Computational Humor*, pages 57–70.

Arthur Schopenhauer. 1883. The world as will and idea (vols. i, ii, & iii). *Haldane, RB, & Kemp, J.(3 Vols.). London: Kegan Paul, Trench, Trubner*, 6.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *NIPS 2015*, pages 2377–2385.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS 2017, Long Beach, CA, USA*, pages 5998–6008.

Orion Weller and Kevin D. Seppi. 2019. Humor detection: A transformer gets the last laugh. In *Proceedings of EMNLP-IJCNLP 2019*, pages 3619–3623.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of ACL 1994*, page 133–138.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard H. Hovy. 2015. Humor recognition and humor anchor extraction. In *Proceedings of EMNLP 2015*, pages 2367–2376.

Dongyu Zhang, Heting Zhang, Xikai Liu, Hongfei Lin, and Feng Xia. 2019. Telling the whole story: A manually annotated chinese dataset for the analysis of humor in jokes. In *Proceedings of EMNLP-IJCNLP 2019*, pages 6401–6406.