SemEval-2017 Task 7: Detection and Interpretation of English Puns

Tristan Miller[*†] and **Christian F. Hempelmann**[†] and **Iryna Gurevych**[*]

[*]Ubiquitous Knowledge Processing Lab (UKP-TUDA/UKP-DIPF)
Department of Computer Science
Technische Universität Darmstadt
https://www.ukp.tu-darmstadt.de

[†]Ontological Semantic Technology Lab
Texas A&M University-Commerce
http://www.tamuc.edu/ontology

Abstract

A pun is a form of wordplay in which a word suggests two or more meanings by exploiting polysemy, homonymy, or phonological similarity to another word, for an intended humorous or rhetorical effect. Though a recurrent and expected feature in many discourse types, puns stymie traditional approaches to computational lexical semantics because they violate their one-sense-per-context assumption. This paper describes the first competitive evaluation for the automatic detection, location, and interpretation of puns. We describe the motivation for these tasks, the evaluation methods, and the manually annotated data set. Finally, we present an overview and discussion of the participating systems' methodologies, resources, and results.

1 Introduction

Word sense disambiguation (WSD), the task of identifying a word's meaning in context, has long been recognized as an important task in computational linguistics, and has been the focus of a considerable number of Senseval/SemEval evaluation tasks. Traditional approaches to WSD rest on the assumption that there is a single, unambiguous communicative intention underlying each word in the document. However, there exists a class of language constructs known as *puns*, in which lexical-semantic ambiguity is a *deliberate* effect of the communication act. That is, the speaker or writer intends for a certain word or other lexical item to be interpreted as simultaneously carrying two or more separate meanings. Though puns are a recurrent and expected feature in many discourse types, they have attracted relatively little attention in the fields of computational linguistics and natural language processing in general, or WSD in particular. In this document, we describe a shared task for evaluating computational approaches to the detection and semantic interpretation of puns.

A pun is a form of wordplay in which one sign (*e.g.,* a word or phrase) suggests two or more meanings by exploiting polysemy, homonymy, or phonological similarity to another sign, for an intended humorous or rhetorical effect (Aarons, 2017; Hempelmann and Miller, 2017). For example, the first of the following two punning jokes exploits the sound similarity between the surface sign "propane" and the latent target "profane", while the second exploits contrasting meanings of the word "interest":

(1) When the church bought gas for their annual barbecue, proceeds went from the sacred to the propane.

(2) I used to be a banker but I lost interest.

Puns where the two meanings share the same pronunciation are known as *homophonic* or *perfect*, while those relying on similar- but not identical-sounding signs are known as *heterophonic* or *imperfect*. Where the signs are considered as written rather than spoken sequences, a similar distinction can be made between *homographic* and *heterographic* puns.

Conscious or tacit linguistic knowledge—particularly of lexical semantics and phonology—is an essential prerequisite for the production and interpretation of puns. This has long made them an attractive subject of study in theoretical linguistics, and has led to a small but growing body of research into puns in computational linguistics. Most computational treatments of puns to date have focused on generative algorithms (Binsted and Ritchie, 1994,

Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 58–68,
Vancouver, Canada, August 3 - 4, 2017.

1997; Ritchie, 2005; Hong and Ong, 2009; Waller et al., 2009; Kawahara, 2010) or modelling their phonological properties (Hempelmann, 2003a,b). However, several studies have explored the detection and interpretation of puns (Yokogawa, 2002; Taylor and Mazlack, 2004; Miller and Gurevych, 2015; Kao et al., 2015; Miller and Turković, 2016; Miller, 2016); the most recent of these focus squarely on computational semantics. In this paper, we present the first organized public evaluation for the computational processing of puns.

We believe computational interpretation of puns to be an important research question with a number of real-world applications. For example:

- It has often been argued that humour can enhance human–computer interaction (HCI) (Hempelmann, 2008), and at least one study (Morkes et al., 1999) has already shown that incorporating canned humour into a user interface can increase user satisfaction without adversely affecting user efficiency. An interactive system that is able to recognize and produce contextually appropriate responses to users' puns could further enhance the HCI experience.

- Recognizing humorous ambiguity is also important in machine translation, particularly for sitcoms and other comedic works, which feature puns and other forms of wordplay as a recurrent and expected feature (Schröter, 2005). Puns can be extremely difficult for non-native speakers to detect, let alone translate. Future automatic translation aids could scan source texts, flagging potential puns for special attention, and perhaps even proposing ambiguity-preserving translations that best match the original pun's double meaning.

- Wordplay is a perennial topic of scholarship in literary criticism and analysis, with entire books (*e.g.,* Wurth, 1895; Rubinstein, 1984; Keller, 2009) having been dedicated to cataloguing the puns of certain authors. Computer-assisted detection and classification of puns could help digital humanists in producing similar surveys of other œuvres.

2 Data sets

The pun processing tasks at SemEval-2017 used two manually annotated data sets, both of which we are freely releasing to the research community.[1]

Our first data set, containing English homographic puns, is based on the one described by Miller and Turković (2016) and Miller (2016).[2] It contains punning and non-punning jokes, aphorisms, and other short, self-contained contexts sourced from professional humorists and online collections. For the purposes of deciding which contexts contain a pun, we used a somewhat weaker definition of homography: the lexical units corresponding to a pun's two distinct meanings must be spelled exactly the same way, with the exception that inflections and particles (*e.g.,* the prepositions or dummy object pronouns in phrasal verbs such as "duke it out") may be disregarded. The contexts have the following characteristics:

- Each context contains a maximum of one pun.

- Each pun (and its latent target) contains exactly one *content word* (*i.e.,* a noun, verb, adjective, or adverb) and zero or more *non-content words* (*e.g.,* prepositions or articles). Here "word" is defined as a sequence of letters delimited by space or punctuation. This means that puns and targets do not include hyphenated words, and they do not consist of multi-word expressions containing more than one content word, such as "get off the ground" or "state of the art". Puns and targets may be multi-word expressions containing only one content word—this includes phrasal verbs such as "take off" or "put up with".

- Each pun (and its target) has a lexical entry in WordNet 3.1. However, the *sense* of the pun or the target may or may not exist in WordNet 3.1.

The homographic data set contains 2250 contexts, of which 1607 (71%) contain a pun. Sense annotation was carried out by three trained human judges, two of whom independently applied sense keys from WordNet 3.1. Each pun word was annotated with two sets of sense keys, one for each meaning of the pun. As in previous Senseval/SemEval word sense annotation tasks, annotators were permitted to select more than one sense key per meaning, or to indicate that the meaning was not listed in

[1] https://www.ukp.tu-darmstadt.de/data/sense-labelling-resources/sense-annotated-english-puns/

[2] The only significant difference is that we removed several hundred of the contexts not containing puns and added them to our new heterographic data set.

pun type	subtask	contexts	words	words / context		
				min	mean	max
homographic	detection	2 250	24 499	2	10.9	44
homographic	location	1 607	18 998	3	11.8	44
homographic	interpretation	1 298	15 510	3	11.9	44
heterographic	detection	1 780	19 461	2	10.9	69
heterographic	location	1 271	15 145	3	11.9	69
heterographic	interpretation	1 098	13 258	3	12.1	69

Table 1: Data set statistics

WordNet. Interannotator agreement, as measured by Krippendorff's (1980) α and a variation of the MASI set comparison metric (Passonneau, 2006; Miller, 2016), was 0.777. Disagreements were resolved automatically by taking the intersection of the corresponding sense sets; for contexts where this was not possible, the third judge manually adjudicated the disagreements. Of the 1607 puns, 1298 (81%) have both meanings in WordNet.

The second data set is similar to the first, except that the puns are heterographic rather than homographic. It was constructed in a similar manner, including the use of two annotators and an adjudicator. However, as heterographic puns have an extra level of complexity (it being sometimes necessary to discuss or explain an obscure joke before one "gets it"), the annotators were given an opportunity to resolve their disagreements themselves before passing the remainder on to the adjudicator. Pre-adjudication agreement for the sense annotations was $\alpha = 0.838$. The final data set contains 1780 contexts, of which 1271 (71%) contain a pun. Of the puns, 1098 (86%) have both meanings in WordNet.

As described in the following section, the two data sets are used in three subtasks—pun detection, pun location, and pun interpretation. The pun detection subtask uses the full data sets, while the other two subtasks use subsets of the full data sets. Table 1 presents some statistics on the size of each subtask's data set in terms of the number of contexts and word tokens.

3 Task definition

Participating systems competed in any or all of the following three subtasks, evaluated consecutively. Within each subtask, participants had the choice of running their system on either or both data sets.

Subtask 1: Pun detection. For this subtask, participants were given an entire raw data set. For each context in the data set, the system had to decide whether or not it contains a pun. For example, take the following two contexts:

(2) I used to be a banker but I lost interest.

(3) What if there were no hypothetical questions?

For (2), the system should have returned "pun", whereas for (3) the system should have returned "non-pun".

Systems had to classify *all* contexts in the data set. Scores were calculated using the standard precision, recall, accuracy, and F-score measures as used in classification (Manning et al., 2008, §8.3):

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F_1 = \frac{2PR}{P + R}$$

where TP, TN, FP, and FN are the numbers of true positives, true negatives, false positives, and false negatives, respectively.

Subtask 2: Pun location. For this subtask, the contexts not containing puns were removed from the data sets. For any or all of the contexts, systems had to make a single guess as to which word is the pun. For example, given context (2) above, the system should have indicated that the tenth word, "interest", is the pun.

Scores were calculated using the standard coverage, precision, recall, and F-score measures as used in word sense disambiguation (Palmer et al., 2007):

$$C = \frac{\text{\# of guesses}}{\text{\# of contexts}}$$

$$P = \frac{\text{\# of correct guesses}}{\text{\# of guesses}}$$

$$R = \frac{\text{\# of correct guesses}}{\text{\# of contexts}}$$

$$F_1 = \frac{2PR}{P+R}.$$

Note that, according to the above definitions, it is always the case that $P \geq R$, and $F_1 = P = R$ whenever $P = R$.

Subtask 3: Pun interpretation. For this subtask, the pun word in each context is marked, and contexts where the pun's two meanings are not found in WordNet are removed from the data sets. For any or all of the contexts, systems had to annotate the two meanings of the given pun by reference to WordNet sense keys. For example, given context (2), the system should have returned the WordNet sense keys `interest%1:09:00::` (glossed as "a sense of concern with and curiosity about someone or something") and `interest%1:21:00::` ("a fixed charge for borrowing money; usually a percentage of the amount borrowed").

As with the pun location subtask, scores were calculated using the coverage, precision, recall, and F-score measures from word sense disambiguation. A guess is considered to be "correct" if one of its sense lists is a non-empty subset of one of the sense lists from the gold standard, and the other of its sense lists is a non-empty subset of the other sense list from the gold standard. That is, the order of the two sense lists is not significant, nor is the order of the sense keys within each list. If the gold standard sense lists contain multiple senses, then it is sufficient for the system to correctly guess only one sense from each list.

4 Baselines

For each subtask, we provide results for various baselines:

Pun detection. The only baseline we use for this subtask is a random classifier. It makes no assumption about the underlying class distribution, labelling each context as "pun" or "non-pun" with equal probability. On average, its recall and accuracy will therefore be 0.5, and its precision equal to the proportion of contexts containing puns.

Pun location. For this subtask we present the results of three naïve baselines. The first simply selects one of the context words at random. The second baseline always selects the last word of the context as a pun. It is informed by empirical studies of large joke corpora, which have found that punchlines tend to occur in a terminal position (Attardo, 1994). The third baseline is a slightly more sophisticated pun location baseline inspired by Mihalcea et al. (2010). In that study, genuine joke punchlines were selected among several non-humorous alternatives by finding the candidate whose words have the highest mean polysemy. We adapt this technique by selecting as the pun the word with the highest polysemy (counting together senses from all parts of speech). In the case of a tie, we choose the most polysemous word nearest to the end of the context.

Pun interpretation. Following the practice in traditional word sense disambiguation, we present the results of the random and most frequent sense baselines, as adapted to pun annotation.

The random baseline attempts to lemmatize the pun word, looks it up in WordNet, and selects two of its senses at random, one for each meaning of the pun. It scores

$$P = R = \frac{1}{n}\sum_{i=1}^{n} \frac{G_1^i \cdot G_2^i}{\binom{S^i}{2}},$$

where n is the number of contexts, G_j^i is the number of gold-standard sense keys in the jth meaning of the pun word in context i, and S^i is the number of sense keys WordNet contains for the pun word in context i. We compute the random baseline only for the homographic data set. (It would in principle be adaptable to the heterographic data set, though the large number of potential target words means the scores would be negligible.)

The most frequent sense (MFS) baseline is a supervised baseline in that it depends on a manually sense-annotated background corpus. As its name suggests, it involves always selecting from the candidates that sense that has the highest frequency in the corpus. For the homographic data set, our MFS implementation attempts to lemmatize the pun word (if necessary, building a list of candidate lemmas) and then selects the two most frequent senses of these lemmas according to WordNet's built-in sense frequency counts.[3] For the heterographic data set, only the first sense is selected from the list of candidate lemmas. A second list is constructed by finding all other lemmas in WordNet

[3]These counts come from the SemCor (Miller et al., 1993) corpus.

with the minimum Levenshtein (1966) distance to the lemmas in the first list. The most frequent sense of the lemmas in the second list is selected as the second meaning of the pun.

In addition to the two naïve baselines, we also provide scores for the homographic pun interpretation system described by Miller and Gurevych (2015). This system works by running each pun through a variation of the Lesk (1986) algorithm that scores each candidate sense according to the lexical overlap with the pun's context. The two top-scoring senses are then selected; in case of ties, the system attempts to select senses which are not closely related to each other, and at least one of whose parts of speech matches the one applied to the pun by a POS tagger.

The baseline pun interpretation scores presented in this paper differ slightly from those given in Miller and Gurevych (2015) and Miller (2016). This is because the scoring program used in those studies compared sense keys on the basis of their underlying WordNet synsets, whereas in this shared task the sense keys are compared directly.

5 Participating systems

Our shared task saw participation from ten systems:

BuzzSaw (Oele and Evang, 2017). BuzzSaw assumes that each meaning of the pun will exhibit high semantic similarity with one and only one part of the context. The system's approach to homographic pun interpretation is to compute the semantic similarity between the two halves of every possible contiguous, binary partitioning of the context, retaining the partitioning with the lowest similarity between the two parts. A Lesk-like WSD algorithm based on word and sense embeddings is then used to disambiguate the pun word separately with respect to each part of the context.

The pun interpretation system is also used for homographic pun location. First, the interpretation system is run once for each polysemous word in the context. The word whose two disambiguated senses have maximum cosine distance between their sense embeddings is selected as the pun word.

Duluth (Pedersen, 2017). For pun detection, the Duluth system assumes that all-words WSD systems will have difficulties in consistently assigning sense labels to contexts containing puns. The system therefore disambiguates each context with four slightly different configurations of the same WSD algorithm. If more than two sense labels differ across runs, the context is assumed to contain a pun. For pun location, the system selects the word whose sense label changed across runs; if multiple words changed senses, then the system selects the one closest to the end of the context.

Homographic pun interpretation is carried out by running various configurations of a WSD algorithm on the pun word and selecting the two most frequently returned senses. For heterographic puns, the system attempts to recover the target form either by generating a list of WordNet lemmas with minimal edit distance to the pun word, or by querying the Datamuse API for words with similar spellings, pronunciations, and meanings. WSD algorithms are then run separately on the pun and the set of target candidates, with the best matching pun and target senses retained.

ECNU (Xiu et al., 2017). ECNU uses a supervised approach to pun detection. The authors collected a training set of 60 homographic and 60 heterographic puns, plus 60 proverbs and famous sayings, from various Web sources. The data is then used to train a classifier, using features derived from WordNet and word2vec embeddings. The ECNU pun locator is knowledge-based, determining each context word's likelihood of being the pun on the basis of the distance between its sense vectors, or between its senses and the context.

ELiRF-UPV (Hurtado et al., 2017). This system's approach to homographic pun location rests on two hypotheses: that the pun will be semantically very similar to one of the non-adjacent words in the sentence, and that the pun will be located near the end of the sentence. The system therefore calculates the similarity between every pair of non-adjacent words in the context using word2vec, retaining the pair with the highest similarity. The word in the pair that is closer to the end of the context is selected as the pun.

To interpret homographic puns, ELiRF-UPV first finds the two context words whose word embeddings are closest to that of the pun.

Then, for each context word, the system builds a bag-of-words representation for each of its candidate senses, and for each of the pun word's candidate senses, using information from WordNet. The lexical overlap between every pair of pun and context senses is calculated, and the pun sense with the highest overlap is selected as one of the meanings of the pun.

Fermi (Indurthi and Oota, 2017). Fermi takes a supervised approach to the detection of homographic puns. Unlike ECNU, the authors did not construct their own data set of puns, but rather split the shared task data set into separate training and test sets, the first of which they manually annotated. A bi-directional RNN then learns a classification model, using distributed word embeddings as input features.

Fermi's approach to pun location is a knowledge-based approach similar to that of ELiRF-UPV. For every pair of words in the context, a similarity score is calculated on the basis of the maximum pairwise similarity of their WordNet synsets. In the highest-scoring pair, the word closest to the end of the context is selected as the pun.

Idiom Savant (Doogan et al., 2017). Idiom Savant uses a variety of different methods depending on the subtask and pun type, but which are generally based on Google *n*-grams and word2vec. Target recovery in heterographic puns involves computing phonetic distance with the aid of the CMU Pronouncing Dictionary. Uniquely among participating systems, Idiom Savant attempts to flag and specially process Tom Swifties, a genre of punning jokes commonly seen in the test data.

JU_CSE_NLP (Pramanick and Das, 2017). As a supervised approach, JU_CSE_NLP relies on a manually annotated data set of 413 puns sourced by the authors from Project Gutenberg. The data is used to train a hidden Markov model and cyclic dependency network, using features from a part-of-speech tagger and a syntactic parser. The classifiers are applied to the pun detection and location subtasks.

PunFields (Mikhalkova and Karyakin, 2017). PunFields uses separate methods for pun detection, location, and interpretation; central to all of them is the notion of semantic fields. The system's approach to pun detection is a supervised one, with features being vectors tabulating the number of words in the context that appear in each of the 34 sections of *Roget's Thesaurus*. For pun location, PunFields uses a weakly supervised approach that scores candidates on the basis of their presence in *Roget's* sections, their position within the context, and their part of speech.

For pun interpretation, the system partitions the context on the basis of semantic fields, and then selects as the first sense of the pun the one whose WordNet gloss has the greatest number of words in common with the first partition. For homographic puns, the second sense selected is the one with the highest frequency count in WordNet (or the next-highest frequency count, in case the first selected sense already has the highest frequency). For heterographic puns, a list of candidate target words is produced using Damerau-Levenshtein (1964) distance. Among their corresponding WordNet senses, the system selects the one whose definition has the highest lexical overlap with the second partition.

UWaterloo (Vechtomova, 2017). UWaterloo is a rule-based pun locator that scores candidate words according to eleven simple heuristics. These heuristics involve the position of the word within the context or relative to certain punctuation or function words, the word's inverse document frequency in a large reference corpus, normalized pointwise mutual information (PMI) with other words in the context, and whether the word exists in a reference set of homophones and similar-sounding words. Only words in the second half of the context are scored; in the event of a tie, the system chooses the word closer to the end of the context.

UWAV (Vadehra, 2017). UWAV participated in the pun detection and location subtasks. The detection component is another supervised system, taking the votes of three classifiers (support vector machine, naïve Bayes, and logistic regression) trained on lexical-semantic and word embedding features of a manually annotated data set.

For pun location, UWAV splits the context in half and checks whether any word in the second half is in some predefined lists of homonyms, homophones, and antonyms. If so, one of those words is selected as the pun. Otherwise, word2vec similarity is calculated between every pair of words in the context. In the highest-scoring word pair, the word closest to the end of the context is selected.

One further team submitted answers after the official evaluation period was over:

N-Hance (Sevgili et al., 2017). The N-Hance system assumes every pun has a particularly strong association with exactly one other word in the context. To detect and locate puns, then, it calculates the PMI between every pair of words in the context. If the PMI of the highest-scoring pair exceeds a certain threshold relative to the other pairs' PMI scores, then the context is assumed to contain a pun, with the pun being the word in the pair closest to the end of the context. Otherwise, the context is assumed to have no pun.

For homographic pun interpretation, the first sense is selected by finding the maximum overlap between the candidate sense definitions and the pun's context. N-Hance then finds the word in the context that has the highest PMI score with the pun. The system selects as the second sense of the pun that sense whose synonyms have the greatest word2vec cosine similarity with the paired word.

6 Results and analysis

Tables 2 through 4 show the results for each of the three subtasks and two data sets. Results for the participating systems are shown in the upper section of each table; the lower section shows the baselines and the N-Hance system entered out of competition. Pun detection results for ECNU and Fermi are also in the non-competition section, since their training data, by accident or design, included some contexts from the test data. To calculate the pun detection scores for these two systems, we first removed the overlapping contexts from the test set.[4] The PunFields pun locator is also marked as it makes use of POS frequency counts of the homographic data set that were published in Miller and Gurevych (2015).

For each metric, the result of the best-performing participating system is shown in boldface. Where a baseline or non-competition entry matched or outperformed the best participating system, its result is also shown in boldface. Generally only the best-scoring run submitted by each system is shown;[5] we have made an exception for Duluth's Datamuse- and edit distance–based pun interpretation variations ("DM" and "ED", respectively), neither of which outperformed the other on all metrics.

Subtask 1: Pun detection. No one system emerged as the clear winner for this subtask, making it hard to draw conclusions on what approaches work best. Among the participating systems for the homographic data set, Punfields achieved the highest precision (0.7993), JU_CSE_NLP the highest recall (0.9079), and Duluth the highest accuracy and F-score (0.7364 and 0.8254, respectively). N-Hance equalled or outperformed the participating systems on recall, accuracy, and F-score. For the heterographic data set, Idiom Savant had the highest precision, accuracy, and F-score (0.8704, 0.7837, and 0.8439, respectively), while JU_CSE_NLP achieved the best recall (0.9402). N-Hance performed about as well as Idiom Savant in terms of F-Score (0.8440). For both data sets, all systems outperformed the random baseline.

Subtask 2: Pun location. The last word baseline (F_1 = 0.4704 and 0.5704 for homographic and heterographic puns, respectively) turned out to be surprisingly hard to beat for this subtask. For the homographic data set, this baseline was exceeded only by Idiom Savant (F_1 = 0.6631) and UWaterloo (F_1 = 0.6523). For the heterographic puns, it was bested only by Idiom Savant (F_1 = 0.6845), UWaterloo (F_1 = 0.7964), and N-Hance (F_1 = 0.6553).

Idiom Savant was not the only system to measure semantic relatedness via word2vec, though it was the only one to do so with *n*-grams from a large background corpus. It was also the only system to directly (albeit simplistically) measure phonetic

[4] Two further supervised pun detection systems, UWAV and Punfields, were found to have inadvertently used training contexts that also appear in the test data. In these two cases, however, the authors removed the overlapping contexts from their training data, retrained their systems, and submitted new results, which we report here.

[5] Participants were permitted to submit the results of up to two runs for each subtask and data set. The intention was to allow participants the opportunity to fix problems in the formatting of their output files, or to try minor variations of the same system.

	homographic				heterographic			
system	**P**	**R**	**A**	**F_1**	**P**	**R**	**A**	**F_1**
Duluth	0.7832	0.8724	**0.7364**	**0.8254**	0.7399	0.8662	0.6871	0.7981
Idiom Savant	—	—	—	—	**0.8704**	0.8190	**0.7837**	**0.8439**
JU_CSE_NLP	0.7251	**0.9079**	0.6884	0.8063	0.7367	**0.9402**	0.7174	0.8261
PunFields	**0.7993**	0.7337	0.6782	0.7651	0.7580	0.5940	0.5747	0.6661
UWAV	0.6838	0.4723	0.4671	0.5587	0.6523	0.4178	0.4253	0.5094
random	0.7142	0.5000	0.5000	0.5882	0.7140	0.5000	0.5000	0.5882
ECNU*	0.7127	0.6474	0.5628	0.6785	0.7807	0.6761	0.6333	0.7247
Fermi†	**0.9024**	0.8970	**0.8533**	**0.8997**	—	—	—	—
N-Hance	0.7553	**0.9334**	**0.7364**	**0.8350**	0.7725	0.9300	0.7545	**0.8440**

Table 2: Pun detection results

	homographic				heterographic			
system	**C**	**P**	**R**	**F_1**	**C**	**P**	**R**	**F_1**
BuzzSaw	**1.0000**	0.2775	0.2775	0.2775	—	—	—	—
Duluth	**1.0000**	0.4400	0.4400	0.4400	**1.0000**	0.5311	0.5311	0.5311
ECNU	**1.0000**	0.3373	0.3373	0.3373	**1.0000**	0.5681	0.5681	0.5681
ELiRF-UPV	**1.0000**	0.4462	0.4462	0.4462	—	—	—	—
Fermi	**1.0000**	0.5215	0.5215	0.5215	—	—	—	—
Idiom Savant	0.9988	**0.6636**	**0.6627**	**0.6631**	**1.0000**	0.6845	0.6845	0.6845
JU_CSE_NLP	**1.0000**	0.3348	0.3348	0.3348	**1.0000**	0.3792	0.3792	0.3792
PunFields‡	**1.0000**	0.3279	0.3279	0.3279	**1.0000**	0.3501	0.3501	0.3501
UWaterloo	0.9994	0.6526	0.6521	0.6523	0.9976	**0.7973**	**0.7954**	**0.7964**
UWAV	**1.0000**	0.3410	0.3410	0.3410	**1.0000**	0.4280	0.4280	0.4280
random	**1.0000**	0.0846	0.0846	0.0846	**1.0000**	0.0839	0.0839	0.0839
last word	**1.0000**	0.4704	0.4704	0.4704	**1.0000**	0.5704	0.5704	0.5704
max. polysemy	**1.0000**	0.1798	0.1798	0.1798	**1.0000**	0.0110	0.0110	0.0110
N-Hance	0.9956	0.4269	0.4250	0.4259	0.9882	0.6592	0.6515	0.6553

Table 3: Pun location results

	homographic				heterographic			
system	**C**	**P**	**R**	**F_1**	**C**	**P**	**R**	**F_1**
BuzzSaw	0.9761	0.1563	**0.1525**	0.1544	—	—	—	—
Duluth (DM)	0.8606	**0.1683**	0.1448	**0.1557**	**0.9791**	0.0009	0.0009	0.0009
Duluth (ED)	0.9992	0.1480	0.1479	0.1480	0.9262	0.0315	0.0291	0.0303
ELiRF-UPV	0.9646	0.1014	0.0978	0.0996	—	—	—	—
Idiom Savant	**0.9900**	0.0778	0.0770	0.0774	0.8434	**0.0842**	**0.0710**	**0.0771**
PunFields	0.8760	0.0484	0.0424	0.0452	0.9709	0.0169	0.0164	0.0166
random	**1.0000**	0.0931	0.0931	0.0931	—	—	—	—
MFS	**1.0000**	0.1348	0.1348	0.1348	**0.9800**	0.0716	0.0701	0.0708
Miller & Gurevych	0.6826	**0.1975**	0.1348	**0.1603**	—	—	—	—
N-Hance	0.9831	0.0204	0.0200	0.0202	—	—	—	—

Table 4: Pun interpretation results

*Evaluated on 2237 of the 2250 homographic contexts, and 1778 of the 1780 heterographic contexts.

†Evaluated on 675 of the 2250 homographic contexts.

‡Uses POS frequency counts from the homographic test set.

distance using a pronunciation dictionary, and the only system that flagged puns of a certain genre for special processing. These features, alone or in combination, may have contributed to the system's success.

UWaterloo and N-Hance were the only systems making use of pointwise mutual information, to which their success might be credited. Evidently the notion of a unique "trigger" word in the context that activates the pun is an important one to model. UWaterloo also shares with Idiom Savant the use of hand-crafted rules based on real-world knowledge of punning jokes.

Subtask 3: Pun interpretation. As in the pun detection subtask, no one approach worked best here, at least for the homographic data set. Only two systems (BuzzSaw and Duluth) were able to beat the most frequent sense baseline. The Miller and Gurevych (2015) system remains the best-performing pun interpreter in terms of precision (0.1975) and F-score (0.1603), though BuzzSaw was able to exceed it in terms of recall (0.1525). Both BuzzSaw and Miller and Gurevych (2015) apply Lesk-like algorithms to "disambiguate" the pun word. However, lexical overlap approaches are also used by most of the lower-performing systems. For heterographic pun interpretation, Idiom Savant achieved the highest scores ($P = 0.0842$, $R = 0.0710$, $F_1 = 0.0771$), though its recall is not much higher than the most frequent sense baseline (0.0701).

It seems that for probabilistic approaches like those submitted, classifying texts as puns and, to a lesser degree, pinpointing the punning lexical material are easier than actual semantic tasks like our Subtask 3. This may be because probabilistic approaches cannot, in principle, see past the arbitrariness of the linguistic sign, instead relying on context to reflect meaning. We assume that producing a full semantic analysis in terms of a knowledge-based system, akin to those proposed in Bar-Hillel's (1960) famous evaluation of fully automatic high-quality translation, might be necessary, because only these approaches can get beyond observed shared features to natural language meaning. Such knowledge-based approaches to meaning in humour, based on relevant semantic humour theories (Raskin, 1985; Attardo and Raskin, 1991), have been in development since Raskin et al. (2009) and one recent (albeit non-scalable) approach, Kao et al. (2015), has already shown very interesting results.

7 Concluding remarks

In this paper we have introduced SemEval-2017 Task 7, the first shared task for the computational processing of puns. We have described the rules for three subtasks—pun detection, pun location, and pun interpretation—and described the manually annotated data sets used for their evaluation. Both data sets are now freely available for use by the research community. We have also described the approaches and presented the results of ten participating teams, as well as several baseline algorithms and a further system entered out of competition.

We observe most systems performed well on the pun detection task, with F-scores in the range of 0.5587 to 0.8440. However, only a few systems beat a simple baseline on pun location. Pun interpretation remains an extremely challenging problem, with most systems failing to exceed the baselines, and with sense assignment accuracy much lower than what is seen with traditional word sense disambiguation. Interestingly, though there exists a considerable body of research in linguistics on phonological models of punning (Hempelmann and Miller, 2017) and on semantic theories of humour (Raskin, 2008), little to none of this work appeared to inform the participating systems.

Acknowledgments

This work has been supported by the German Institute for Educational Research (DIPF). The authors thank Edwin Simpson for helping build the heterographic data set.

References

Debra Aarons. 2017. Puns and tacit linguistic knowledge. In Salvatore Attardo, editor, *The Routledge Handbook of Language and Humor*, Routledge, New York, NY, Routledge Handbooks in Linguistics, pages 80–94.

Salvatore Attardo. 1994. *Linguistic Theories of Humor*, Mouton de Gruyter, Berlin, chapter 2: The Linear Organization of the Joke. https://doi.org/10.1515/9783110219029.

Salvatore Attardo and Victor Raskin. 1991. Script theory revis(it)ed: Joke similarity and joke representation model. *Humor: International Journal of Humor Research* 4(3–4):293–348. https://doi.org/10.1515/humr.1991.4.3-4.293.

Yehoshua Bar-Hillel. 1960. The present status of automatic translation of languages. In Franz L. Alt,

editor, *Advances in Computers*, Academic Press, volume 1, pages 91–163.

Kim Binsted and Graeme Ritchie. 1994. An implemented model of punning riddles. In *Proceedings of the Twelfth National Conference on Artificial Intelligence: AAAI-94*. pages 633–638.

Kim Binsted and Graeme Ritchie. 1997. Computational rules for generating punning riddles. *Humor: International Journal of Humor Research* 10(1):25–76. https://doi.org/10.1515/humr.1997.10.1.25.

Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM* 7(3):171–176. https://doi.org/10.1145/363958.363994.

Samuel Doogan, Aniruddha Ghosh, Hanyang Chen, and Tony Veale. 2017. Idiom Savant at SemEval-2017 Task 7: Detection and interpretation of English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 103–108.

Christian F. Hempelmann. 2003a. *Paronomasic Puns: Target Recoverability Towards Automatic Generation*. Ph.D. thesis, Purdue University, West Lafayette, IN.

Christian F. Hempelmann. 2003b. YPS – The Ynperfect Pun Selector for computational humor. In *Proceedings of the CHI 2003 Workshop on Humor Modeling in the Interface*.

Christian F. Hempelmann. 2008. Computational humor: Beyond the pun? In Victor Raskin, editor, *The Primer of Humor Research*, Mouton de Gruyter, Berlin, number 8 in Humor Research, pages 333–360. https://doi.org/10.1515/9783110198492.333.

Christian F. Hempelmann and Tristan Miller. 2017. Puns: Taxonomy and phonology. In Salvatore Attardo, editor, *The Routledge Handbook of Language and Humor*, Routledge, New York, NY, Routledge Handbooks in Linguistics, pages 95–108.

Bryan Anthony Hong and Ethel Ong. 2009. Automatically extracting word relationships as templates for pun generation. In *Computational Approaches to Linguistic Creativity: Proceedings of the Workshop*. pages 24–31.

Lluís-F. Hurtado, Encarna Segarra, Ferran Pla, Pascual Andrés Carrasco Gómez, and José Ángel González. 2017. ELiRF-UPV at SemEval-2017 Task 7: Pun detection and interpretation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 439–442.

Vijayasaradhi Indurthi and Subba Reddy Oota. 2017. Fermi at SemEval-2017 Task 7: Detection and interpretation of homographic puns in English language. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 456–459.

Justine T. Kao, Roger Levy, and Noah D. Goodman. 2015. A computational model of linguistic humor in puns. *Cognitive Science* 40(5):1270–1285. https://doi.org/10.1111/cogs.12269.

Shigeto Kawahara. 2010. Papers on Japanese imperfect puns. Online collection of previously published journal and conference articles. http://user.keio.ac.jp/~kawahara/pdf/punbook.pdf.

Stefan Daniel Keller. 2009. *The Development of Shakespeare's Rhetoric: A Study of Nine Plays*. Number 136 in Swiss Studies in English. Narr, Tübingen.

Klaus Krippendorff. 1980. *Content Analysis: An Introduction to Its Methodology*. Number 5 in The Sage CommText Series. Sage Publications, Beverly Hills, CA.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice cream cone. In Virginia DeBuys, editor, *SIGDOC '86: Proceedings of the 5th Annual International Conference on Systems Documentation*. pages 24–26. https://doi.org/10.1145/318723.318728.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10(8):707–710.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge.

Rada Mihalcea, Carlo Strapparava, and Stephen Pulman. 2010. Computational models for incongruity detection in humour. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing: 11th International Conference, CICLing 2010*. Springer, Berlin/Heidelberg, number 6008 in Theoretical Computer Science and General Issues, pages 364–374. https://doi.org/10.1007/978-3-642-12116-6_30.

Elena Mikhalkova and Yuri Karyakin. 2017. PunFields at SemEval-2017 Task 7: Employing *Roget's Thesaurus* in automatic pun recognition and interpretation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 425–430.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Human Language Technology: Proceedings of a Workshop Held at Plainsboro, New Jersey*. San Francisco, CA, pages 303–308. https://doi.org/10.3115/1075671.1075742.

Tristan Miller. 2016. *Adjusting Sense Representations for Word Sense Disambiguation and Automatic Pun Interpretation*. Dr.-Ing. thesis, Department of Computer Science, Technische Universität Darmstadt.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of English puns. In *The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing: Proceedings of the Conference*. volume 1, pages 719–729.

Tristan Miller and Mladen Turković. 2016. Towards the automatic detection and identification of English puns. *European Journal of Humour Research* 4(1):59–75.

John Morkes, Hadyn K. Kernal, and Clifford Nass. 1999. Effects of humor in task-oriented human–computer interaction and computer-mediated communication: A direct test of SRCT theory. *Human–Computer Interaction* 14(4):395–435. https://doi.org/10.1207/S15327051HCI1404_2.

Dieke Oele and Kilian Evang. 2017. BuzzSaw at SemEval-2017 Task 7: Global vs. local context for interpreting and locating homographic English puns with sense embeddings. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 443–447.

Martha Palmer, Hwee Tou Ng, and Hoa Trang Dang. 2007. Evaluation of wsd systems. In Eneko Agirre and Philip Edmonds, editors, *Word Sense Disambiguation: Algorithms and Applications*, Springer, number 33 in Text, Speech, and Language Technology, chapter 4, pages 75–106.

Rebecca J. Passonneau. 2006. Measuring agreement on set-valued items (MASI) for semantic and pragmatic annotation. In *5th Edition of the International Conference on Language Resources and Evaluation*. pages 831–836.

Ted Pedersen. 2017. Duluth at SemEval-2017 Task 7: Puns upon a midnight dreary, lexical semantics for the weak and weary. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 384–388.

Aniket Pramanick and Dipankar Das. 2017. JU_CSE_NLP at SemEval-2017 Task 7: Employing rules to detect and interpret English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 431–434.

Victor Raskin. 1985. *Semantic Mechanisms of Humor*. Number 24 in Studies in Linguistics and Philosophy. Springer Netherlands. https://doi.org/10.1007/978-94-009-6472-3.

Victor Raskin, editor. 2008. *The Primer of Humor Research*. Number 8 in Humor Research. Mouton de Gruyter, Berlin. https://doi.org/10.1515/9783110198492.

Victor Raskin, Christian F. Hempelmann, and Julia M. Taylor. 2009. How to understand and assess a theory: The evolution of SSTH into the GTVH and now into the OSTH. *Journal of Literary Theory* 3(2):285–312. https://doi.org/10.1515/JLT.2009.016.

Graeme D. Ritchie. 2005. Computational mechanisms for pun generation. In Graham Wilcock, Kristiina Jokinen, Chris Mellish, and Ehud Reiter, editors, *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG-05)*. pages 125–132.

Frankie Rubinstein. 1984. *A Dictionary of Shakespeare's Sexual Puns and Their Significance*. Macmillan, London.

Thorsten Schröter. 2005. *Shun the Pun, Rescue the Rhyme? The Dubbing and Subtitling of Language-play in Film*. Ph.D. thesis, Karlstad University.

Özge Sevgili, Nima Ghotbi, and Selma Tekir. 2017. N-Hance at SemEval-2017 Task 7: A computational approach using word association for puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 435–438.

Julia M. Taylor and Lawrence J. Mazlack. 2004. Computationally recognizing wordplay in jokes. In Kenneth Forbus, Dedre Gentner, and Terry Regier, editors, *Proceedings of the Twenty-sixth Annual Conference of the Cognitive Science Society*. pages 1315–1320.

Ankit Vadehra. 2017. UWAV at SemEval-2017 Task 7: Automated feature-based system for locating puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 448–451.

Olga Vechtomova. 2017. UWaterloo at SemEval-2017 Task 7: Locating the pun using syntactic characteristics and corpus-based metrics. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 420–424.

Annalu Waller, Rolf Black, David A. O'Mara, Helen Pain, Graeme Ritchie, and Ruli Manurung. 2009. Evaluating the STANDUP pun generating software with children with cerebral palsy. *ACM Transactions on Accessible Computing* 1(3):1–27. https://doi.org/10.1145/1497302.1497306.

Leopold Wurth. 1895. *Das Wortspiel bei Shakspere*. Wilhelm Braumüller, Vienna.

Yuhuan Xiu, Man Lan, and Yuanbin Wu. 2017. ECNU at SemEval-2017 Task 7: Using supervised and unsupervised methods to detect and locate English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 452–455.

Toshihiko Yokogawa. 2002. Japanese pun analyzer using articulation similarities. In *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems: FUZZ 2002*. volume 2, pages 1114–1119. https://doi.org/10.1109/FUZZ.2002.1006660.