# N-Hance System Recreation for SemEval-2017 Task 7

**Cameron Knopp** and **Jorge Mazariegos**

## Abstract

We recreated the N-Hance system as described in the paper by Sevgili et. al. in order to provide a possible solution for SemEval-2017 Task 7. We achieved results substantially lower than those achieved by the original system, though we believe this is mainly due to a difference in how our PMI scores were generated, among some other slight differences which we describe.

## 1 Introduction

Word sense disambiguation (WSD) is the task of identifying a word's meaning in context. This task has long been recognized as an important task of natural language processing, and has been the focus of many other SemEval tasks. When working with WSD, traditional approaches rest on the assumption that there is a single, unambiguous communicative intention underlying each word in the document. Puns are a type of language construct that are deliberately designed to be ambiguous in conveying their intent. This deliberate ambiguity presents a challenge to the traditional approaches of WSD and requires a more novel approach.

Task 7 of the International Workshop on Semantic Evaluation is an attempt to tackle this issue head-on as participants are allowed to solve the problem in whichever way they choose. The task consists of 3 subtasks: pun detection, pun location and pun interpretation.

### 1.1 Subtask I (Description)

For this subtask, systems are given a .xml file of sentences. For each sentence, a given system must output either '1' or '0' to indicate whether or not the sentence contains a pun.

### 1.2 Subtask II (Description)

For this subtask, systems are once again given a .xml file of sentences, though this time sentences not containing puns are omitted. The system must parse each sentence and identify the word ID of the pun word, according to the word IDs specified in the .xml file.

### Subtask III (Description)

For this subtask, the system is given a .xml file of sentences along with the word ID of the pun word in each sentence. With this information, the system must output the first two intended senses of the pun word according to the WordNet sense keys.

## 2 Related Work

Including the N-Hance system, there were a total of eleven entries for this SemEval task, although N-Hance was a late submission. Each of the submissions employed a variety of different approaches for solving these subtasks. The next nine subsections are brief descriptions of the other submissions, followed by tables containing the final results of the competition on all three subtasks.

### 2.1 BuzzSaw

BuzzSaw (Oele and Evang, 2017) assumes that each meaning of the pun will exhibit high semantic similarity with one and only one part of the context.

### 2.2 Duluth

Duluth (Pedersen, 2017) assumes that all words WSD systems will have difficulties in consistently assigning sense labels to contexts containing puns.

### 2.2 ECNU

ECNU (Xiu et al., 2017) makes no assumptions and uses a supervised approach to pun detection on a rather small dataset.

### 2.3 ELiRF-UPV

ELiRF-UPV (Hurtado et al., 2017) is based upon 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.

### 2.4 Fermi

Fermi (Indurthi and Oota, 2017) is similar to ELiRF-UPV. Fermi takes a supervised approach to the detection of puns but does not construct their own dataset of puns as ECNU did. Fermi instead splits the shared task data set into separate training and test sets, the first of which they manually annotate.

### 2.5 Idiom Savant

Idiom Savant (Doogan et al., 2017) combines a variety of methods generally based on Google n-grams and word2vec.

### 2.6 JU_CSE_NLP

JU_CSE_NLP (Pramanick and Das, 2017) is another supervised approach relying on a manually annotated dataset of 413 puns sourced by the authors from Project Gutenberg.

### 2.7 PunFields

PunFields (Mikhalkova and Karyakin, 2017) uses three separate methods for each subtask that all rely on the notion of semantic fields.

### 2.8 UWaterloo

UWaterloo (Vechtomova, 2017) is a rule-based pun locator based on eleven simple heuristics.

### 2.8 UWAV

UWAV (Vadehra, 2017) only participated in the first two subtasks of detection and location.

### 2.9 N-Hance

The N-Hance system assumes that every pun appears at the end of a sentence, and has a particularly strong association with exactly one other word in the context. The system focuses more on homographic puns over heterographic puns and interprets them based on finding the maximum overlap between candidate sense definitions and the pun's context.

### 2.10 Results of Competition

The following tables show the original results of the SemEval 2017 Task 7.

| | 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 1: Pun Detection competition 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 2: Pun Location competition 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 3: Pun Interpretation competition results.

## 3 Datasets

The following subsections describe the three datasets used by our system.

### 3.1 Heterographic Puns Dataset

The heterographic puns dataset provided by the SemEval competition contains 1780 English-language sentences, of which 1271 contain a heterographic pun. This is one of the datasets that the system is scored on.

Heterographic puns are puns that rely on different senses of two words which have a similar sound, rhyme, and/or close spelling. An example of this kind of pun is:

*"A dentist hates having a bad day at the orifice."*

In this example, the pun word is 'orifice', which stands-in for the word 'office'. Without knowledge of this word-swap, this sentence is rather nonsensical, and that is the nature of heterographic puns.

### 3.2 Homographic Puns Dataset

The homographic puns dataset provided by the SemEval competition contains 2250 English-language sentences, of which 1607 contain a pun. This is one of the datasets that the system is scored on.

Homographic puns are characterized by the multiple meanings a word can have that lead to different but still valid interpretations of the sentence. An example of a homographic pun is:

*"Getting rid of your boat for another could cause a whole raft of problems."*

In this example, the pun word is 'raft', since the sentence is about boats. While 'raft' can refer to a boat of some kind, it can also refer to a large quantity of something, which is the intended sense of the word in this case.

### 3.3 wiki-news-300d-1M

This dataset contains one million word vectors trained on Wikipedia 2017, the UMBC WebBase corpus, and the stamt.org news dataset. We used this dataset in subtask III to calculate cosine similarity between words. The original N-Hance system creators trained their own word2vec model on Wikipedia data, but we thought that it would make more sense for us, due to computational limitations and the widespread availability of such models, to use one of these premade models.

## 4 Subtask Solutions

In the following subsections, we provide descriptions of the methodology we followed in an attempt to solve each of the three subtasks. Our methodology follows that which is described in the N-Hance paper.

### 4.1 Data Preparation

The datasets provided from the SemEval committee are in .xml format and need to be processed before we can use them. Using Beautiful Soup 4 we extract the raw text from the .xml files and store them into various lists and dictionaries to be used in all of our subtask solutions.

### 4.2 Subtask I (Solution)

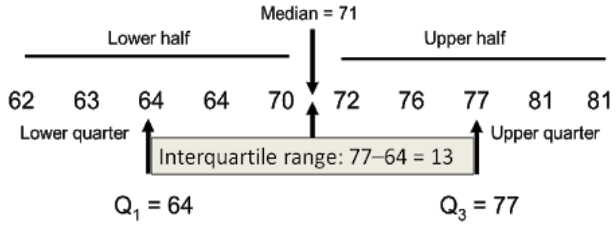
For this subtask, we begin by removing stop words and generating Skipgram word pairs for each of the input sentences. The Skipgram window size is set to the length of the given sentence so that every possible word pair is generated for the given sentence. Following this, the reverse of each word pair is pruned. This is done so that later on there are no repeats in PMI scores generated for each word pair.

Next, we generate PMI scores for each word pair in each sentence. In this scenario, PMI, Pointwise Mutual Information, measures the probability of the two words co-occurring, given their context, with the following formula:

$$pmi(w1, w2) = log_2 \frac{p(w1, w2)}{p(w1)p(w2)}$$

The PMI scores are generated using NLTK bigram collocations created from the input sentences with a window size of 20.

After a PMI score has been generated for every word pair in every sentence, the interquartile range of the PMI scores for each sentence is found. The interquartile range is calculated using the scipy.stats iqr function, which finds the interquartile range in the following manner:




Now that there is an interquartile range assigned to each sentence, the median of all of these ranges is found.

For a given sentence, if the highest PMI score minus the second-highest PMI score is greater than the median interquartile range, then that sentence is marked as containing a pun. The pseudocode below illustrates this idea:

```
If highest_pmi_score − second_highest_pmi_score > median_iqr:
    sentence_contains_pun = True
```

The median interquartile range is used here as a global threshold value by which PMI scores can be measured against. If a word pair's PMI score is notably high, as indicated by comparison with this threshold value, then that indicates the presence of a pun word.

### 4.3 Subtask II (Solution)

For this subtask, we first remove stop words and generate Skipgram word pairs and PMI scores in the same manner as described in the previous subtask. For each sentence, we conclude that the pun word is the second word in the word pair with the highest PMI score.

We reach this conclusion since the pun word most frequently occurs at the end of a given sentence. Since we previously removed the reverse of each word pair, only word pairs in which the second word is located later in the sentence than the first word remain. For instance, given the example pun, "'I'm halfway up a mountain,' Tom alleged," we find that the highest PMI score occurs between the words "Tom" and "alleged". We then rightfully conclude that "alleged" is the pun word, since it is the second word in the pair and therefore occurs closer to the end of the sentence than 'Tom'.

### 4.4 Subtask III (Solution)

For this subtask, we find the first sense of the given pun word using pywsd's simple lesk algorithm. This algorithm takes in a word and its context (the full sentence) and returns a WordNet sense key indicating the intended dictionary definition of the word.

Next, we begin our task of finding the second sense of the pun word by retrieving the highest PMI score pair to which the pun word belongs. In the example from the previous subtask, this word pair would be (Tom, alleged). We take the non-pun word in this word pair

(e.g., “Tom”) and refer to it as the target word. We then gather all synonyms, in the form of WordNet sense keys, of the pun word using NLTK WordNet synsets.

For each pun word synonym, we find the cosine similarity between the synonym and the target word. We do this using the ‘wiki-news-300d-1M’ word2vec model, which we have described in the datasets section of this paper. We load in this model in the special .magnitude format for increased efficiency and speed using the Pymagnitude package. This additionally allows for the quick calculation of word vectors for out-of-vocabulary words, which is especially useful in this context.

After finding the cosine similarity between each synonym and the target word, we conclude that the synonym with the highest cosine similarity to the target word is the second sense of the pun word.

## 5 System Output

For each subtask, our system reads in an .xml file and produces output to a .txt file. The following subsections provide screenshots of our system’s output along with brief explanations of their content.

### 5.1 Subtask I Output

In the following figure, we provide screenshots from our system’s subtask I output. On each line, the system outputs the sentence ID, as provided by the .xml file, along with either a ‘1’ or a ‘0’ to indicate whether or not the sentence contains a pun.

```
het_1 0
het_2 1
het_3 1
het_4 1
het_5 1
het_6 0
het_7 0
het_8 1
het_9 1
het_10 0
het_11 0
het_12 0
het_13 0
het_14 0
het_15 0
het_16 0
het_17 1
het_18 1
het_19 0
het_20 0
het_21 1
het_22 1
```

```
hom_1 0
hom_2 0
hom_3 0
hom_4 0
hom_5 0
hom_6 0
hom_7 0
hom_8 0
hom_9 0
hom_10 0
hom_11 0
hom_12 0
hom_13 1
hom_14 0
hom_15 0
hom_16 0
hom_17 1
hom_18 0
```

Figure 1: Screenshot of system output for Subtask I heterographic (left) and homographic (right).

### 5.2 Subtask II Output

In the following figure, we provide screenshots of our system’s output for subtask II. On each line, the system outputs the given sentence ID, followed by a space, and then finally the word ID of the pun word in that sentence.

```
het_1 het_1_14
het_2 het_2_11
het_4 het_4_8
het_5 het_5_3
het_7 het_7_3
het_8 het_8_8
het_9 het_9_2
het_11 het_11_13
het_12 het_12_4
het_13 het_13_27
het_15 het_15_3
het_16 het_16_3
het_17 het_17_3
het_19 het_19_7
het_20 het_20_16
het_22 het_22_6
het_23 het_23_13
het_24 het_24_6
het_25 het_25_9
het_26 het_26_8
```

```
hom_1 hom_1_11
hom_2 hom_2_10
hom_3 hom_3_3
hom_4 hom_4_5
hom_5 hom_5_12
hom_7 hom_7_14
hom_8 hom_8_10
hom_9 hom_9_7
hom_10 hom_10_5
hom_11 hom_11_3
hom_14 hom_14_6
hom_15 hom_15_4
hom_18 hom_18_4
hom_19 hom_19_4
hom_20 hom_20_8
hom_21 hom_21_10
hom_22 hom_22_9
hom_23 hom_23_5
hom_25 hom_25_6
hom_26 hom_26_5
hom_28 hom_28_3
```

Figure 2: Screenshot of system output for Subtask II heterographic (left) and homographic (right).

### 5.3 Subtask III Output

In the following figures, we provide screenshots of our system's output for subtask III. On each line, the system outputs the word ID of the pun word, as given by the .xml file, followed by a space. The system outputs the first sense of the pun word, a space, and then the second sense of the pun word. The pun word senses are in the form of WordNet sense keys, which refer to a specific dictionary definition of that word, since a given word can have many different definitions.

```
het_1_14 allege%2:32:00:: supposed%5:00:00:questionable:00
het_2_13 coolly%4:02:00:: jack%1:18:01::
het_4_11 orifice%1:08:00:: clarence_shepard_day_jr.%1:18:00::
het_5_5 gnu%1:05:00:: wildebeest%1:05:00::
het_7_6 harry%2:30:00:: barber%2:29:00::
het_9_6 step%1:06:00:: repugn%2:32:00::
het_11_13 archly%4:02:00:: archly%4:02:00::
het_12_12 hertz%1:28:00:: die%2:30:02::
het_13_24 dye%2:30:00:: accidentally%4:02:02::
het_15_3 sleep_together%2:35:00:: know%2:31:14::
het_16_11 maid%1:18:00:: bird%1:18:00::
het_19_16 octave%1:10:01:: power%1:09:01::
het_22_17 punctually%4:02:00:: pin%2:33:00::
het_24_6 psalm%2:36:00:: psalm%2:36:00::
het_25_7 thickly%4:02:04:: well-worn%5:00:00:unoriginal:00
```

Figure 3: Screenshot of system output for Subtask III heterographic.

```
hom_3_7 sting%2:37:00:: abuse%2:30:02::
hom_4_5 impinge%2:41:10:: shovel%1:06:02::
hom_5_15 forge%2:38:01:: in_the_lead%4:02:00::
hom_7_14 hit%1:04:02:: hit%2:32:12::
hom_8_17 vault%2:38:00:: worn_out%5:00:00:tired:00
hom_9_11 cryptic%5:00:00:inexplicable:00 jockey%2:33:01::
hom_10_13 pin%1:06:01:: clasp%2:35:03::
hom_11_9 solve%2:31:00:: cream%2:30:00::
hom_14_14 admit%2:42:03:: admit%2:42:03::
hom_15_15 toast%2:34:00:: party%2:41:00::
hom_16_13 plant%2:31:00:: handler%1:18:00::
hom_18_8 square%1:25:00:: geometry%1:09:00::
hom_19_3 dangerous%5:00:00:critical:03 situation%1:04:00::
hom_20_17 wall_socket%1:06:00:: paint_a_picture%2:32:00::
hom_21_9 absorb%2:35:00:: draw%2:29:00::
hom_22_15 take_a_hit%2:34:12:: disperse%2:35:00::
hom_23_7 combustion%1:22:00:: question%2:32:02::
hom_26_15 go%1:28:00:: disaster%1:04:00::
hom_27_10 bounce%2:40:01:: jump_shot%1:04:00::
```

Figure 4: Screenshot of system output for Subtask III homographic.

## 6 Scoring and Results

The scorer, provided by the competition, is run separately for each subtask and input dataset. The scorer compares our system's output, in the form of a .txt file, to the .gold file containing the correct answers for each subtask. It outputs the system's results on several metrics, including precision, accuracy, recall, f1, and coverage (the used metrics change based on the subtask).

In the following subsections we display tables with our results for each subtask alongside the results achieved by the original N-Hance system. 'Het.' and 'Hom." refer to the

heterographic and homographic puns datasets, respectively, and 'P' and 'R' refer to precision and recall, respectively.

### 6.1 Subtask I Results

| Type | P | R | F1 |
|---|---|---|---|
| Het. | .7488 | .5885 | .6590 |
| Hom. | .7528 | .3335 | .4622 |

Table 4: Our results for Subtask I.

| Type | P | R | F1 |
|---|---|---|---|
| Het. | .7725 | .9300 | .8440 |
| Hom. | .7553 | .9334 | .8350 |

Table 5: N-Hance results for Subtask I.

### 6.2 Subtask II Results

| Type | P | R | F1 |
|---|---|---|---|
| Het. | .2269 | .1330 | .1677 |
| Hom. | .2530 | .1319 | .1734 |

Table 6: Our results for Subtask II.

| Type | P | R | F1 |
|---|---|---|---|
| Het. | .7725 | .9300 | .8440 |
| Hom. | .6592 | .6515 | .6553 |

Table 7: N-Hance results for Subtask II.

### 6.3 Subtask III Results

| Type | P | R | F1 |
|---|---|---|---|
| Het. | 9.5329E-4 | 9.1075E-4 | 9.3153E-4 |
| Hom. | .0032 | .0031 | .0031 |

Table 8: Our results for Subtask III.

| Type | P | R | F1 |
|---|---|---|---|
| Het. | - | - | - |
| Hom. | .0204 | .0200 | .0202 |

Table 9: N-Hance results for Subtask III. Heterographic results omitted by creators due to excessively poor results.

## 7 Conclusion

From our recreation of the N-Hance system and reviewing the work of the other participants, we conclude that current state of the art systems are unable to solve the problem of pun interpretation. No system submitted, including our own recreation, was able to get a passable score for the third subtask.

We believe a large part of the difficulty of this task for machines is that they only have access to the written form of pun words and must rely on strict definitions and synonyms of these words to try and piece together the intended meaning of the word. Any current machine-based approach is limited by only being able to access this single dimension of information. A human, on the other hand, can both read the word and hear it said aloud, making the task of pun interpretation generally trivial.

Furthermore, we partially attribute the generally poor results achieved by all systems to

the questionable quality of the provided pun datasets. Upon reviewing the given datasets, we find a rather wide range of quality, going from good puns such as,

*"two construction workers had a stairing contest"*

to more questionable puns, such as,

*"'I'd like to be a Chinese laborer,' said Tom coolly."*

Both sentences are said by the contest organizers to contain a pun, but we were unable to recognize where the pun was located in the second sentence upon hearing and reading it. If a human cannot even recognize the sentence as containing a pun, then will a machine recognize it? Clearly, the contest creators had to resort to making-up their own puns after a certain point, since there were around 3,000 puns in both datasets combined. This would explain the low quality of quite a few of the sentences in the datasets.

## 8 Future Works

We believe that one of the main causes of our non-optimal results (in comparison to those achieved by N-Hance) was the manner in which we generated PMI scores. While the original N-Hance system generated PMI scores using NLTK bigram collocations on a Wikipedia dataset plus the given puns datasets, we only generated the bigram collocations on the given pun datasets. This would result in a less accurate word probability calculation in the PMI score formula. We would need more computing resources in order to accomplish this, though, because our machines were not able to handle the creations of such a large number of bigram collocations.

Another possible reason why our scores were non-optimal is the fact that some of the pun words were not actual words, but rather intentional misspellings, such as 'haulways'. This would result in those words not appearing when searched in WordNet sense keys and therefore our system would not generate any results for that word on subtask 3. So, with more time and computing resources we would create much larger bigram collocations in order to more accurately calculate PMI scores, and we would also try to find the closest word to a given misspelled word, so that it would return a result if searched up in WordNet. Additionally, we would test out a variety of methods from all of the other submitted systems, and create a single system that utilizes the best parts of every other system.

## 9 References


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.

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.

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.

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

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.

Miller, T., Hempelmann, C., & Gurevych, I. (2017). SemEval-2017 Task 7: Detection and Interpretation of English Puns. Proceedings of the 11th International Workshop on Semantic Evaluation. doi: 10.18653/v1/s17-2005

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.

Sevgili, Ö., Ghotbi, N., & Tekir, S. (2017). N-Hance at SemEval-2017 Task 7: A Computational Approach using Word Association for Puns. Proceedings of the 11th International Workshop on Semantic Evaluation. doi: 10.18653/v1/s17-2074

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.

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.

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.