Modern Approaches to the Sentence-Completion Task

CS224U Literature Review

Bonnie Nortz (bnortz@stanford.edu)

Tyler Romero (tromero1@stanford.edu)

***Abstract*—We review and compare papers that present solutions to the sentence completion problem. We find that models using pointwise mutual information (PMI), recurrent neural networks (RNNs), dependency parsing, and latent semantic analysis (LSA) are all approaches to this problem that have been successful to some extent. In addition, we compare the datasets and evaluation metrics that are commonly used for the sentence completion problem.**

Keywords—sentence completion, near-synonyms, SAT, GRE, fill-in-the-blank, PMI, LSA, RNNs, context, connotation

I. Motivation

Sentence completion questions are designed to evaluate skilled reading ability. Skilled reading is a challenging problem that requires accurate parsing of the structure of sentences and the connotations of words. High school and college aged students are evaluated on their verbal reasoning ability through sentence completion questions on the Scholastic Aptitude Test (SAT) and Graduate Readiness Exam (GRE), respectively. For artificial semantic models, these types of questions can also serve a purpose, perhaps as a more accurate measure of their performance than current methods of evaluation such as perplexity, which does not always correlate with downstream success. And besides measuring overall progress, high performance on sentence completion tasks can prove a system's ability in subtasks of machine translation and natural language generation, where a system must choose the most appropriate words while paying attention to lexical nuances.

II. General Problem/Task Definition

A. Formal Definition

We begin by formalizing the sentence completion problem, so that we can draw comparisons between automatic methods for attempting sentence completion.

A test sentence $t = \{w_1, ..., w_{j-1}, s_i, w_{j+1}, ..., w_m\}$ contains a word s_i which belongs to a set of candidate words $S = \{s_1, ..., s_n\}$, $1 \leq i \leq n$. A test case is created by removing s_i from t, and replacing it with a symbol representing a blank space. We call the resulting sentence the context c. The context c and the set of possible answers S are presented to subjects, which then are asked to guess which of the words in S is the missing word.

The papers we evaluate have different takes on this general format. Some papers allow for there to be two missing words in the same passage. Some require that S is made up of near synonyms to s_i (for instance, $S = \{error, mistake, oversight\}$), and others allow S to be composed of more or less unrelated words (e.g. $S = \{systematic, voluntary, experimental, clinical\}$).

The set of papers we have reviewed use various datasets for evaluation, which have slightly different implementations of sentence completion questions. For the papers involving near-synonyms, they generally evaluate their models using a dataset drawn from Wall Street Journal (WSJ) articles written in 1989, that has around 30,000 examples, with 7 different sets of near synonyms. These include sets like "mistake, error, fault", and "job,

task, chore" [3]. For each test example, the system is presented with a sentence with a word missing, and the list of near synonyms (the size of which ranges from 2 to 4 words), and it must choose the exact one that was taken out in order for the example to be considered correct, even if more than one could reasonably complete the sentence.

Recently, Microsoft Research (MSR) released a sentence completion dataset and an associated competition [8], which includes 1,040 sentences with a single word removed, taken from *Sherlock Holmes* novels. For each of these, the system is presented with the sentence and a set of 5 candidate words, which are unrelated, and again it must choose the single word that correctly completes the sentence.

Finally, for any paper that involves completing SAT questions, these will usually be of the same format as the MSR sentences, with one word missing and 5 unrelated candidate words provided. However, the SAT also has sentence completion questions involving two blanks that must be filled at once; some papers explicitly say that they take on these types of questions, and for others it is unclear whether they do or do not.

B. Evaluation Metrics

Since sentence completions questions are either answered correctly or incorrectly, they are scored on the basis of accuracy.

Evaluation becomes more interesting and complex in the case of standardized tests like the SAT and GRE. For example, on the SAT, subjects are awarded 1 point for a correct answer, 0 points for a blank answer, and -0.25 points for an incorrect answer, to penalize low-confidence guessing. Therefore achieving a good SAT score is more complex than just choosing what seems to be the best answer. Having a concept of confidence may be useful in reducing incorrect answers.

In addition, by evaluating on an actual SAT or GRE test, our model can be very accurately compared to human performance on this task. Using test scores, we can find the percentile among human test takers in which our model placed.

C. Sources of Training Data

As mentioned above, SAT and GRE tests that have been made public are a great for evaluating sentence completion models. However, they do not provide enough questions to be useful as a training corpus.

Thankfully, it is easy to generate training corpora by coming up with a list of vocabulary words (large lists of this type have been created for students practicing for the SAT), coming up with n synonyms for each word, and then scanning complex and involved sources of text, such as the New York Times or 19th century works of literature, and creating training examples by removing vocab words from the text.

In addition, since this type of task can be used to evaluate more general semantic models, such models could be trained on full sentences, not just ones that have words missing. Instead of strictly being classifiers predicting which candidate word should fill the blank, these types of models could score the likelihood of any particular sentence, and choose the most likely of all candidates. As mentioned above, sources such as the New York Times or 19th century literature would be potential datasets, with little preprocessing necessary. Also, along with the MSR sentence completion evaluation set, Microsoft Research provides a corresponding training set that includes 522 19th century novels from Project Gutenberg.

III. Paper Summaries

A. A Statistical Model for Near-Synonym Choice (Inkpen, 2007) [4]

This paper explores three approaches to automatic sentence completion. The first, pointwise mutual information (PMI), is an information theoretic take on the problem. The pointwise mutual information between two words x and y is a ratio comparing the probability of observing the two

words together to the probabilities of observing them independently.

$$PMI(x,y) = log_2[P(x,y)/P(x)P(y)]$$

These probabilities can be approximated by $P(x) = C(x)/N$, $P(y) = C(y)/N$, $P(x,y) = C(x,y)/N$. Where C denotes frequency counts and N is the total number of words in the corpus.

A score is generated by taking a window of size k on either side of a missing word and computing $Score(v,s) =$

$\sum_{j=1}^{k} PMI(v,w_j) + \sum_{j=k+1}^{2k} PMI(v,w_j)$, for each $v \in S$. The word v that maximizes the score is selected as the answer.

This paper also proposes using anti-collocations to eliminate incorrect answer choices. An anti-collocation is a combination of words that a native speaker would not use. This method ranks the strongest collocations higher than the anti collocations. And answers based on these rankings.

Finally, the paper puts forth a supervised learning approach. Sentences are featurized using the 500 most frequent words situated close to the missing words in a training set. The value of a word feature for each example is 1 if the word is present and 0 otherwise. Naive Bayes and Boosted Decision Trees achieved the highest accuracy using these features.

Results were trained on the Waterloo Multitext System, and evaluated on the WSJ 1989 corpus.

B. Near-Synonym Choice Using a 5-gram Language Model (Islam and Inkpen, 2010) [2]

The researchers in this paper use the Google Web 1T n-gram corpus and a backoff 5-gram language model, basically calculating the likelihood of a sentence using the number of times its constituent n-grams appear in the dataset. They perform some slightly more detailed statistical modeling with the n-gram counts to handle the interaction between different sized n-grams. For instance, they try to make sure that if a 5-gram in a sentence is very frequent, its value is not overestimated if it also contains a very frequent 4-gram.

To evaluate the model, this paper uses the WSJ dataset of near-synonym sentence completion examples.

C. Near-Synonym Lexical Choice in Latent Semantic Space (Wang and Hirst, 2010) [3]

In this paper, a latent semantic space is built by using SVD to perform dimensionality reduction on a word-document co-occurrence matrix. This results in a vector representation for each word in the corpus.

These vectors can be leveraged in an unsupervised model by using typical distance functions such as cosine similarity between a near-synonym's word vector and the context represented by a weighted centroid of words. The nearest neighbor algorithm is then run to identify the best word for the context.

In addition, they implement a supervised learning model using these latent semantic space features. A support vector machine is used to transform the weighted context centroid into a classification.

These researchers train and evaluate on the same datasets as Islam and Inkpen, 2010.

D. Computational Approaches to Sentence Completion (Zweig et al., 2010) [7]

Working with both SAT questions and the Microsoft Research sentence completion dataset, this paper tests three types of models: n-gram language models (both Good-Turing smoothing and maximum-entropy), a recurrent neural network (RNN) language model, and latent semantic analysis (LSA). They also use an ensemble of these models to get their final results, as they hoped to make use of both the local information captured by the n-gram and RNN models and the global context captured by the LSA model.

In order to combine the models, the paper proposes a simple linear combination of the results of all models. LSA contributes total word similarity,

the cosine similarity between sum of the solution word vectors and the sum of the word vectors of the rest of the sentence, and the number of terms in the answer that are not in the vocabulary. The other models contribute the probability of each candidate sentence according to the model. Finally, the weights of the linear combination are trained on the development set for the SAT task, or the training data for the Microsoft Research task, where they attempt to minimize a pairwise loss function with one correct answer and one incorrect answer from the same question.

To train their models for the SAT task, the researchers use data from the LA times spanning from 1985 to 2002, containing 1.1B words. As discussed above, the MSR challenge comes with its own training data - 522 novels from Project Gutenberg containing 48M words.

For evaluation, the researchers use a set of 11 practice SAT test, with 5 tests (95 questions) selected for the development set and 6 tests (108 questions) for the test set. For the MSR task, they use the provided set of 1,040 questions.

E. Dependency Recurrent Neural Language Models for Sentence Completion (Mirowski and Vlachos, 2015) [6]

Looking to improve on previous deep learning models, this team combines RNNs with dependency information about the word being predicted and the surrounding sentence. Each sentence goes through dependency parsing, which creates a parse tree with one root, many leaves, and with each leaf possessing a single path from the root to itself, called an "unroll". The paper proposes a simple modification to training the RNN, where on each unroll, it is reset and run independently. And, since some words will appear in more unrolls than others, each word token is given a weight discount inversely proportional to how many unrolls it appears in. Finally, an additional matrix connecting label features to output word probabilities allows them to use the dependency labels as well as the dependency structure.

The researchers also add a maximum-entropy model to the RNNs by creating a matrix that connects the n-gram context of an input word to the output word probabilities. They find that this extension adds significantly to the performance of the classic RNN model as well as their dependency RNN.

This paper used the data from the MSR sentence-completion challenge, so they trained on the 522 novels from Project Gutenberg, and split the 1,040 test sentences into 520 development examples and 520 test examples.

F. Exploiting Linguistic Features for Sentence Completion (Woods, 2016) [5]

This paper presents a model that builds upon the PMI model put forth in previous papers. In this model, context sentences are preprocessed by removing determiners, coordinating conjunctions, pronouns, and proper nouns. The remaining words were then added to a feature set. Words in the context that share a syntactic dependency with a candidate word are included in another feature set. Finally, sentence keywords were identified and placed into a feature set. These words then make up the new 'context' upon which each candidate word is scored. The PMI metric is augmented into positive PMI (PPMI) by the following formula:

$$PPMI(x,y) = max(0,\ PMI(x,y))$$

The candidate word that achieves the highest similarity score is selected as the most likely answer.

Bigrams and trigrams were also incorporated into the feature sets described above.

This paper evaluates on both MSR and SAT data; for the MSR evaluation, the typical Project Gutenberg training data and 1,040 testing examples were used. For the SAT task, the training data was the New York Times portion of the Gigaword corpus, and the testing data included 285 sentence completion problems from practice SAT exams given by the College Board, the creator of the SAT.

G. Taking the SAT Exam: Sentence Completion (Busch et al., 2012) [1]

A former Stanford CS224U paper, the authors propose three models for answering SAT questions, generally using the New York Times section of the Gigaword corpus as training data. They first build a trigram and 5-gram language model from their training data, using Good-Turing discounting and Katz backoff. Next, they attempt the information theoretic approach of using positive PMI, with the score of a candidate being the sum of all pairwise PPMI scores between words in the candidate answer and the rest of the sentence, assigning a pairwise PPMI score of 0 if they encountered any word not in their vocabulary.

Finally, they implement a dictionary method, where they use information from WordNet to help inform their choice. For each candidate word, they find its WordNet gloss, the words in its synset, and the words in any direct relation to it in WordNet, and use the Porter stemmer to preprocess all of these words. After preprocessing the words in the question sentence in the same way, they calculate the number of words that overlap in these two sets, and choose the candidate answer with the highest amount of overlap.

The team also tried to combine the results from all of their methods by various means: training a logistic regression with the output scores from each model to predict the answer, generating confidence scores for each model and summing them equally, and with a weighted sum. However, none of these results out-performed their individual PPMI model in the end.

As a validation set, these researchers use 408 sentence completion questions from various online test-preparation sources, which provide practice SAT questions. In order to evaluate their work, they use a test set of 190 practice SAT questions published by the creators of the SAT themselves.

IV. Results and Comparison

A. N-gram Methods

Many papers we reviewed implemented some form of an n-gram language model; some implemented only this model, and others combined information from n-grams with other models, including LSA and RNNs. Islam and Inkpen (2010) use a 5-gram language model on its own, and Busch et al. (2012) used a trigram and 5-gram language model on its own and in combination with their other PPMI and dictionary-based models (although neither of these ended up being their highest performing system). Zweig et al. (2010) successfully used a Good-Turing n-gram model in conjunction with LSA and RNNs in an ensemble, and Mirowski and Vlachos (2015) used a trigram max-entropy model along with dependency RNNs. Finally, Woods (2016) also includes bigrams and trigrams in the "context features" portion of her model.

Each of these papers realize that the local context of the missing word in the sentence contains a wealth of information about the word itself, and just the statistics of how frequently certain combinations of these local contexts appear in a corpus can be a powerful predictor, even in the case of near synonymy with Islam and Inkpen. That being said, four of these five groups attempted to combine this highly localized information with more global context, with PPMI, LSA, RNNs, and/or syntactic dependency information. Zweig et al., Mirowski and Vlachos, and Woods benefited greatly from the combination, and while Busch et al. did not achieve quite as good results with the ensemble methods they tried, they proposed that with a better combination technique they might have been able to.

B. PMI Methods

PMI methods represent the current state of the art on the MSR and SAT datasets, achieving accuracies of 0.6144 and 0.5895 respectively. For PMI based models, we do not have to retrain for each near-synonym group or set of candidate answers, which also makes them more flexible than n-gram based models. In addition, PMI models are

computationally efficient and and straightforward to implement.

Paper	Reference	Approach	Data Set	Accuracy
Zweig, et al. (2010)	[7]	LSA Total Similarity	MSR	0.49
		LSA Total Similarity	SAT	0.46
		LSA + Good-Turing Language Model +RNN	MSR	0.52
		LSA + Good-Turing Language Model	SAT	0.53
Busch, Katherine, et al. (2012)	[1]	PPMI	SAT	0.463
Mirowski and Vlachos (2015)	[6]	Dependency RNNs + 3-gram Max-Entropy	MSR	0.527
Woods (2016)	[5]	PPMI + context features	**MSR**	**0.6144**
		PPMI + context features	**SAT**	**0.5895**
Inkpen (2007)	[4]	PMI	WSJ 1989	0.66
Islam and Inkpen (2010)	[2]	5-Gram Language Model	WSJ 1989	0.699
Wang and Hirst (2010)	[3]	SVMs on Latent Vectors	**WSJ 1989**	**0.779**

Table 1: Results of the reviewed papers.
(Note: The *SAT* datasets presented in the table are all slightly different, as they were gathered by different research groups, but we thought the results would be roughly comparable.)

The most basic PMI implementation was put forth in Inkpen (2007). Other PMI models build directly upon this paper. Woods et al. modified this approach to use positive PMI, as well as used a significant amount of additional feature engineering in order to modify the computation of the final score for each word. These changes had a profound effect on the performance of the model. Finally, Busch et al. applied PPMI to answer SAT questions, without additional feature engineering, which underperforms when compared to Woods et al.

C. Sentence Dependencies

Some researchers decided to incorporate syntactic dependency information into their models, largely with success. Mirowski and Vlachos used dependency parsing in addition to RNNs in their highest performing system, and Woods included words that shared dependencies with the candidate word as a set of features in her model.

In the case of Mirowski and Vlachos, the dependency information encoded in the model was quite detailed, as the RNNs were trained on labeled parse trees, while for Woods' model, the information was much more simplistic, as it was just that the syntactically related words were placed in their own feature set. However, based on the success of both of these models on both SAT and MSR tasks, it is clear that capturing the structured global context of the entire sentence can be quite valuable to a model.

D. SAT Question Data/Obscure words

One of the main challenges faced by sentence-completion models is dealing with obscure words. As noted by Busch et al. and Woods, answers to SAT questions tend to not appear frequently in the training corpus, making them difficult to deal with. To counter this, Zweig et al. had a specific feature in their LSA model that counts the number of terms in the answer that aren't in the vocabulary.

V. Future Work

A. GRE Questions

So far, there have only been attempts to apply these approaches to SAT questions, but, to our knowledge, no one has evaluated these models on GRE questions. GRE questions are meant for college students as opposed to high school students. Owing to this, GRE fill-in-the-blank questions are more challenging, and incorporate vocabulary that is more obscure. It would be interesting to see if these models hold up on more difficult questions.

B. Sentence Equivalence Tasks

A closely related, but more challenging task is sentence equivalence, which can be found in questions on the GRE. Given the sentence and the candidate words, exactly two words must be chosen such that the sentence is completed correctly and has

the same meaning. These types of questions combine the challenge of sentence completion with some of the challenges of near-synonymy, since the candidate solution words are not all unrelated as they typically are in the MSR and SAT sentence completion tasks.

C. Improved Generalization

A common problem encountered by models attempting to solve SAT problems is that the candidate solutions tend to be obscure, so they occasionally do not appear in the training vocabulary. Utilizing WordNet synsets, as well as hypernyms, to capture more commonly known semantic information about words could be used to help solve this problem. In addition, incorporating morphological structure into the language models (teaching the model root words, prefixes, and suffixes), could help the model to generalize better to obscure words.

D. Experimental Improvements

Busch et al. (2012) use practice SAT questions from non-College Board sources as validation data and only College Board practice SAT questions as test data, so it is difficult to say how much of their difficulty to generalize from validation to test data was the quality of their model or the differing quality of practice SAT questions. In order to remedy this, one could form the training, test, and development sets with equal parts official and unofficial questions, so that the quality of the questions is not as much of a confounding variable.

VI. References

[1] Busch, Katherine, et al. *Taking the SAT Exam: Sentence Completion*. 2012.

[2] Islam, and Inkpen. *Near-Synonym Choice Using a 5-Gram Language Model.* University of Ottawa. 2010.

[3] Wang, and Hirst. *Near-Synonym Lexical Choice in Latent Semantic Space.* 2010.

[4] Inkpen. *A Statistical Model for Near-Synonym Choice.* 2007.

[5] Woods. *Exploring Linguistic Features for Sentence Completion.* 2016.

[6] Mirowski, and Vlachos. *Dependency Recurrent Neural Language Models for Sentence Completion.* 2015.

[7] Zweig, et al. *Computational Approaches to Sentence Completion.* 2010.

[8] Zweig, and Burges. "The Microsoft Research Sentence Completion Challenge." Microsoft Research Technical Report, 20 Feb. 2011.