Title: Paraphrasing (computational linguistics)

URL: https://en.wikipedia.org/wiki/Paraphrasing_(computational_linguistics)

PageID: 56142183

Categories: Category:Computational linguistics, Category:Machine learning

Source: Wikipedia (CC BY-SA 4.0).

Paraphrase or paraphrasing in computational linguistics is the natural language processing task of detecting and generating paraphrases . Applications of paraphrasing are varied including information retrieval, question answering , text summarization , and plagiarism detection . [1] Paraphrasing is also useful in the evaluation of machine translation , [2] as well as semantic parsing [3] and generation [4] of new samples to expand existing corpora . [5]

Paraphrase generation

Multiple sequence alignment

Barzilay and Lee [5] proposed a method to generate paraphrases through the usage of monolingual parallel corpora , namely news articles covering the same event on the same day. Training consists of using multi-sequence alignment to generate sentence-level paraphrases from an unannotated corpus. This is done by

finding recurring patterns in each individual corpus, i.e. " X (injured/wounded) Y people, Z seriously" where X, Y, Z are variables

finding pairings between such patterns the represent paraphrases, i.e. " X (injured/wounded) Y people, Z seriously" and " Y were (wounded/hurt) by X , among them Z were in serious condition"

This is achieved by first clustering similar sentences together using n-gram overlap. Recurring patterns are found within clusters by using multi-sequence alignment. Then the position of argument words is determined by finding areas of high variability within each cluster, aka between words shared by more than 50% of a cluster's sentences. Pairings between patterns are then found by comparing similar variable words between different corpora. Finally, new paraphrases can be generated by choosing a matching cluster for a source sentence, then substituting the source sentence's argument into any number of patterns in the cluster.

Phrase-based machine translation

Paraphrase can also be generated through the use of phrase-based translation as proposed by Bannard and Callison-Burch. [6] The chief concept consists of aligning phrases in a pivot language to produce potential paraphrases in the original language. For example, the phrase "under control" in an English sentence is aligned with the phrase "unter kontrolle" in its German counterpart. The phrase "unter kontrolle" is then found in another German sentence with the aligned English phrase being "in check," a paraphrase of "under control."

The probability distribution can be modeled as Pr (e 2 | e 1) {\displaystyle \Pr(e_{2}|e_{1})} , the probability phrase e 2 {\displaystyle e_{2}} is a paraphrase of e 1 {\displaystyle e_{1}} , which is equivalent to Pr (e 2 | f) Pr (f | e 1) {\displaystyle \Pr(e_{2}|f)\Pr(f|e_{1})} summed over all f {\displaystyle f} , a potential phrase translation in the pivot language. Additionally, the sentence e 1 {\displaystyle e_{1}} is added as a prior to add context to the paraphrase. Thus the optimal paraphrase, e 2 ^ {\displaystyle {\hat {e_{2}}}} can be modeled as:

Pr (e 2 | f) {\displaystyle \Pr(e_{2}|f)} and Pr (f | e 1) {\displaystyle \Pr(f|e_{1})} can be approximated by simply taking their frequencies. Adding S {\displaystyle S} as a prior is modeled by calculating the probability of forming the S {\displaystyle S} when e 1 {\displaystyle e_{1}} is substituted with e 2 {\displaystyle e_{2}} .

Long short-term memory

There has been success in using long short-term memory (LSTM) models to generate paraphrases. [7] In short, the model consists of an encoder and decoder component, both implemented using variations of a stacked residual LSTM. First, the encoding LSTM takes a one-hot encoding of all the words in a sentence as input and produces a final hidden vector, which can represent the input sentence. The decoding LSTM takes the hidden vector as input and generates a new sentence, terminating in an end-of-sentence token. The encoder and decoder are trained to take a phrase and reproduce the one-hot distribution of a corresponding paraphrase by minimizing perplexity using simple stochastic gradient descent . New paraphrases are generated by inputting a new phrase to the encoder and passing the output to the decoder.

Transformers

With the introduction of Transformer models , paraphrase generation approaches improved their ability to generate text by scaling neural network parameters and heavily parallelizing training through feed-forward layers . [8] These models are so fluent in generating text that human experts cannot identify if an example was human-authored or machine-generated. [9] Transformer-based paraphrase generation relies on autoencoding , autoregressive , or sequence-to-sequence methods. Autoencoder models predict word replacement candidates with a one-hot distribution over the vocabulary, while autoregressive and seq2seq models generate new text based on the source predicting one word at a time. [10] [11] More advanced efforts also exist to make paraphrasing controllable according to predefined quality dimensions, such as semantic preservation or lexical diversity. [12] Many Transformer-based paraphrase generation methods rely on unsupervised learning to leverage large amounts of training data and scale their methods. [13] [14]

Paraphrase recognition

Recursive autoencoders

Paraphrase recognition has been attempted by Socher et al [1] through the use of recursive autoencoders . The main concept is to produce a vector representation of a sentence and its components by recursively using an autoencoder. The vector representations of paraphrases should have similar vector representations; they are processed, then fed as input into a neural network for classification.

Given a sentence W {\displaystyle W} with m {\displaystyle m} words, the autoencoder is designed to take 2 n {\displaystyle n} -dimensional word embeddings as input and produce an n {\displaystyle n} -dimensional vector as output. The same autoencoder is applied to every pair of words in S {\displaystyle S} to produce ■ m / 2 ■ {\displaystyle \lfloor m/2\rfloor } vectors. The autoencoder is then applied recursively with the new vectors as inputs until a single vector is produced. Given an odd number of inputs, the first vector is forwarded as-is to the next level of recursion. The autoencoder is trained to reproduce every vector in the full recursion tree, including the initial word embeddings.

Given two sentences W 1 {\displaystyle W_{1}} and W 2 {\displaystyle W_{2}} of length 4 and 3 respectively, the autoencoders would produce 7 and 5 vector representations including the initial word embeddings. The euclidean distance is then taken between every combination of vectors in W 1 {\displaystyle W_{1}} and W 2 {\displaystyle W_{2}} to produce a similarity matrix S ∈ R 7 × 5 {\displaystyle S\in \mathbb {R} ^{7\times 5}} . S {\displaystyle S} is then subject to a dynamic min-pooling layer to produce a fixed size n p × n p {\displaystyle n_{p}\times n_{p}} matrix. Since S {\displaystyle S} are not uniform in size among all potential sentences, S {\displaystyle S} is split into n p {\displaystyle n_{p}} roughly even sections. The output is then normalized to have mean 0 and standard deviation 1 and is fed into a fully connected layer with a softmax output. The dynamic pooling to softmax model is trained using pairs of known paraphrases.

Skip-thought vectors

Skip-thought vectors are an attempt to create a vector representation of the semantic meaning of a sentence, similarly to the skip gram model . [15] Skip-thought vectors are produced through the use of a skip-thought model which consists of three key components, an encoder and two decoders. Given a corpus of documents, the skip-thought model is trained to take a sentence as input and encode it into a skip-thought vector. The skip-thought vector is used as input for both

decoders; one attempts to reproduce the previous sentence and the other the following sentence in its entirety. The encoder and decoder can be implemented through the use of a recursive neural network (RNN) or an LSTM .

Since paraphrases carry the same semantic meaning between one another, they should have similar skip-thought vectors. Thus a simple logistic regression can be trained to good performance with the absolute difference and component-wise product of two skip-thought vectors as input.

Transformers

Similar to how Transformer models influenced paraphrase generation, their application in identifying paraphrases showed great success. Models such as BERT can be adapted with a binary classification layer and trained end-to-end on identification tasks. [16] [17] Transformers achieve strong results when transferring between domains and paraphrasing techniques compared to more traditional machine learning methods such as logistic regression . Other successful methods based on the Transformer architecture include using adversarial learning and meta-learning . [18] [19]

Evaluation

Multiple methods can be used to evaluate paraphrases. Since paraphrase recognition can be posed as a classification problem, most standard evaluations metrics such as accuracy , f1 score , or an ROC curve do relatively well. However, there is difficulty calculating f1-scores due to trouble producing a complete list of paraphrases for a given phrase and the fact that good paraphrases are dependent upon context. A metric designed to counter these problems is ParaMetric. [20] ParaMetric aims to calculate the precision and recall of an automatic paraphrase system by comparing the automatic alignment of paraphrases to a manual alignment of similar phrases. Since ParaMetric is simply rating the quality of phrase alignment, it can be used to rate paraphrase generation systems, assuming it uses phrase alignment as part of its generation process. A notable drawback to ParaMetric is the large and exhaustive set of manual alignments that must be initially created before a rating can be produced.

The evaluation of paraphrase generation has similar difficulties as the evaluation of machine translation . The quality of a paraphrase depends on its context, whether it is being used as a summary, and how it is generated, among other factors. Additionally, a good paraphrase usually is lexically dissimilar from its source phrase. The simplest method used to evaluate paraphrase generation would be through the use of human judges. Unfortunately, evaluation through human judges tends to be time-consuming. Automated approaches to evaluation prove to be challenging as it is essentially a problem as difficult as paraphrase recognition. While originally used to evaluate machine translations, bilingual evaluation understudy (BLEU) has been used successfully to evaluate paraphrase generation models as well. However, paraphrases often have several lexically different but equally valid solutions, hurting BLEU and other similar evaluation metrics. [21]

Metrics specifically designed to evaluate paraphrase generation include paraphrase in n-gram change (PINC) [21] and paraphrase evaluation metric (PEM) [22] along with the aforementioned ParaMetric. PINC is designed to be used with BLEU and help cover its inadequacies. Since BLEU has difficulty measuring lexical dissimilarity, PINC is a measurement of the lack of n-gram overlap between a source sentence and a candidate paraphrase. It is essentially the Jaccard distance between the sentence, excluding n-grams that appear in the source sentence to maintain some semantic equivalence. PEM, on the other hand, attempts to evaluate the "adequacy, fluency, and lexical dissimilarity" of paraphrases by returning a single value heuristic calculated using N-grams overlap in a pivot language. However, a large drawback to PEM is that it must be trained using large, in-domain parallel corpora and human judges. [21] It is equivalent to training a paraphrase recognition to evaluate a paraphrase generation system.

The Quora Question Pairs Dataset, which contains hundreds of thousands of duplicate questions, has become a common dataset for the evaluation of paraphrase detectors. [23] Consistently reliable paraphrase detection have all used the Transformer architecture and all have relied on large amounts of pre-training with more general data before fine-tuning with the question pairs.

See also

Round-trip translation

Text simplification – Automated process

Text normalization – Process of transforming text into a single canonical form

References

External links

Microsoft Research Paraphrase Corpus - a dataset consisting of 5800 pairs of sentences extracted from news articles annotated to note whether a pair captures semantic equivalence

Paraphrase Database (PPDB) - A searchable database containing millions of paraphrases in 16 different languages