Evaluating Contrastive Learning Strategies for Enhanced Performance in Downstream Tasks

Stanford CS224N Default Project

Zachary Behrman
Department of Computer Science
Stanford University
`behrmanz@stanford.edu`

Georgios Christoglou
Department of Computer Science
Stanford University
`gglou@stanford.edu`

Abstract

Sentence embeddings are an important, but challenging domain of NLP. Recent developments with self-attention, transformers, and BERT have allowed for a new learned embedding paradigm. While these methods have made significant strides, our paper aims to survey the recent advancements of contrastive learning on embedding quality. Within this paper, we first aim to implement unsupervised SimCSE, supervised SimCSE and DiffCSE, alongside extensions using data augmentation. With these pretraining strategies defined, we additionally develop classification heads for textual similarity, sentiment analysis, and paraphrase detection. Finally, We assess each pretraining strategy on the downstream classification heads, with the goal of finding the most performant architecture. Through our experimentation, we found that the author's version of supervised SimCSE + our developed classification heads performed best, placing us on the top $\sim \frac{1}{3}$ of leaderboard submissions.

1 Key Information to include

- Mentor: Anirudh Sriram
- Contributions: Both Zach and George contributed implementations, experiments, and writing. There is no discernable difference to report.

2 Introduction

Within the constantly-evolving NLP field, sentence embeddings have remained an incredibly central, but difficult topic of exploration. For example, "The movie left her feeling terrible" differs by $\sim$1 word from "The movie left her feeling terribly excited", yet the meaning conveyed is almost completely different. While humans can understand these nuances, how can we convey such information to computers, who have no underlying concept of language to fall-back on?

While the problem is difficult, improvements to embedding robustness affect large-scale systems like Google search (Kobayashi and Takeda, 2000), as well as more concrete NLP tasks such as semantic textual similarity, paraphrase detection, and text classifications.

The recent advents of self-attention and the transformer architecture (Vaswani et al., 2023) have enabled a more efficient and scalable training paradigm. This was built further upon by BERT (Devlin et al., 2019) which defines an efficient pre-training scheme to generate robust embeddings. Finally, recent developments such as SimCSE (Gao et al., 2022a) and DiffCSE (Chuang et al., 2022) expand upon BERT pretraining with new contrastive learning strategies (discussed further in the Related Work section).

Within this paper, we first aim to compare the vanilla BERT, SimCSE, and DiffCSE pretraining strategies, while proposing a new "augmented" Supervised SimCSE extension. Second, we develop

3 task-specific classification heads on top of our embedding system to properly perform semantic textual similarity, paraphrase detection, and sentiment analysis.

After experimentation and analysis, we found the original author's implementation of SimCSE performed the best upon semantic textual similarity, paraphrase detection, and sentiment analysis. We believe this to be a indication of the importance on quality for supervised contrastive approaches.

3 Related Work

3.1 BERT

BERT Devlin et al. (2019) introduced a significant advancement within NLP by leveraging the Transformer architecture to pre-train deep bidirectional representations from unlabeled text. This is achieved through the use of a "masked language model" (MLM) pre-training objective, where random tokens are masked out and the model is trained to predict them solely on their context. This enables a deepr understanding of language context and relationships. By pre-training on a large corpus and then fine-tuning for specific objectives, BERT has achieved SOTA in multiple downstream tasks including sentiment analysis, text similarity and others.

3.2 Anisotropy and Contrastive Learning

Work by Ethayarajh (2019) suggested that learned sentence embbedings often suffered from an underlying anisotropy problem. That is, embeddings take on a tight conal shape in vector space (as opposed to a more uniformal distribution). This tight shape restriction could thus affect the expressiveness of resultant embeddings.

Within the computer-vision space, work by Chen et al. (2020) had made significant strides to combat anisotropy through a contrastive learning framework. Under contrastive learning, semanticaly similar neighbors are brought closer together while semantically dissimilar neighbors are spread farther apart. Within N mini-batches, contrastive learning operates under the following objective function:

$$-\log \frac{e^{sim(h_i,h_j)/\tau}}{\sum_{j=1}^{N} e^{sim(h_i,h_j)/\tau}} \tag{1}$$

Where (h_i, h_j) are hidden state representations of semantically similar inputs (x_i, x_j).

Specifically, Chen et al. (2020) found that semantically similar pairs (x_i, x_j) could be synthetically created through data augmentation.

3.3 Unsupervised SimCSE

SimCSE (Gao et al., 2022a) was a recent development aimed at applying the contrastive learning methodology to combat the anisotropy problem within sentence embeddings. Particularly, for every input x_i, an associated x_i^+ was created through dropout. The pair (x_i, x_i^+) were then encoded into (h_i, h_i^+) and used to optimize the contrastive loss (equation 1).

3.4 Supervised SimCSE

Additionally, SimCSE devised a strategy for a supervised framework utilizing Natural Language Inference (NLI) datasets. Under this premise, a tuple (x_i, x_i^+, x_i^-) are generated, where x_i^+ is a separate sentence entailed to x_i and x_i^- is a seperate sentence contradictory to x_i. This tuple is then utilized in a modified contrastive learning objective function:

$$\mathcal{L}_{supsimcse} = -\log \frac{e^{\frac{sim(h_i,h_t^+)}{\tau}}}{\sum_{j=1}^{N} \left(e^{\frac{sim(h_i,h_j^+)}{\tau}} + e^{\frac{sim(h_i,h_j^-)}{\tau}} \right)} \tag{2}$$

With the goal that entailed h_i, h_i^+ pairs will occur closer together while contradictory h_i, h_i^- pairs will be pushed apart.

3.5 DiffCSE

DiffCSE (Chuang et al., 2022) is an extension upon the SimCSE contrastive learning paradigm. Particularly, DiffCSE aims to introduce a separate, related task to the overall training framework. While the contrastive learning objective remains the same, DiffCSE introduces an additional difference prediction task with use of a discriminator neural network.

At a high-level , for every input sentence x, an additional x' is created with probabilistic masking applied. x' is subsequently fed into a small, fixed (i.e without gradient updates) generator model in order to generate mask word predictions labeled x''.

x'' is subsequently encoded and fed into the discriminator alongside the resultant embedding from the unsupervised SimCSE process. The discriminator model will then use the embeddings of x'' and SimCSE's embedding of x to determine whether each token was replaced. The resultant loss function of DiffCSE therefore takes into account both the contrastive loss from SimCSE and the cross-entropy loss from the discriminator D's replacement predictions:

$$\begin{aligned}
\mathcal{L}_{diffcse} &= \mathcal{L}_{contrast} + \mathcal{L}_{discrim} && (3)\\
&= -\log \frac{e^{sim(h_i, h_i^+)/\tau}}{\sum_{j=1}^{N} e^{sim(h_i, h_j^+)/\tau}} && (4)\\
&\quad + \sum_{t=1}^{T} \Big(-\mathbf{1}(x''_t = x_t) \log D(x'', h, t) && (5)\\
&\quad - \mathbf{1}(x''_t \neq x_t) \log(1 - D(x'', h, t)) \Big) && (6)
\end{aligned}$$

4 Approach

4.1 Pretraining Strategies

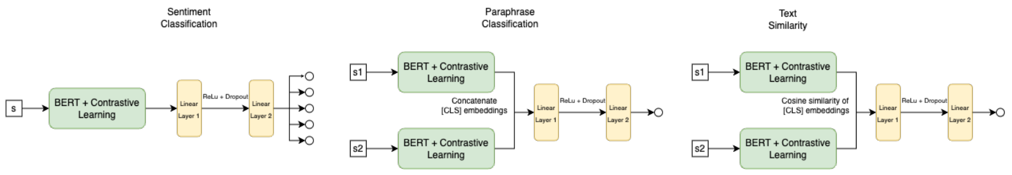
In order to pick the optimal pretraining strategy for downstream performance, we decided to create our own implementations of BERT, Unsupervised SimCSE, Supervised SimCSE, and DiffCSE (more information on these pretraining strategies can be found in the Related Work section).

We aim to deepen our understanding of the SimCSE model's learning dynamics under different training conditions. Specifically, we investigate the effects of varying training data sizes and the integration of additional training samples through dataset augmentation. These efforts are directed towards optimizing the model's performance and exploring the synergy between supervised and unsupervised learning paradigms. Below, we detail the experimental setups employed in our investigation:

- **Unsupervised SimCSE Experimentation:** To assess how much data is needed for the unsupervised SimCSE to effectively learn, experiments were conducted using different sizes of the training data. This approach helps in understanding the impact of dataset size on the model's learning efficiency.
- **Supervised SimCSE Experimentation:** Experiments were conducted using two datasets for the supervised SimCSE approach:
    1. The original dataset curated by the authors. Gao et al. (2022b)
    2. An augmented version of the dataset, which adds contradiction pairs from the MultiNLI dataset. Since each pair being added does not have a corresponding entailment pair, we apply dropout to create a synthetic entailed input. This is inspired by unsupervised SimCSE's use of dropout for introducing variability. We hope this will combine the strengths of both supervised and unsupervised approaches. As an end result, we add an additional $105k$ training samples to the dataset.

4.2 Classification Heads

For classification, we have chosen to utilize task-specific classification heads on top of our pretrained BERT models, the details of which can be seen below.


Sentiment Classification
s
BERT + Contrastive Learning
Linear Layer 1
ReLu + Dropout
Linear Layer 2
Paraphrase Classification
s1
s2
BERT + Contrastive Learning
BERT + Contrastive Learning
Concatenate [CLS] embeddings
Linear Layer 1
ReLu + Dropout
Linear Layer 2
Text Similarity
s1
s2
BERT + Contrastive Learning
BERT + Contrastive Learning
Cosine similarity of [CLS] embeddings
Linear Layer 1
ReLu + Dropout
Linear Layer 2

Final model design for the default project's 3 downstream tasks.

Semantic Textual Similarity: For the STS task, we first calculate the cosine similarity between the sentence embeddings of $s1$ and $s2$ and we feed the value to a two-layer MLP with ReLU activation. (first layer of [1, 10] and the second [10, 1]).

Sentiment Classification For sentiment classification, sentence s is first encoded into representative embeddings with BERT, before being fed into a 2-layer MLP with ReLU activation (first layer of shape [768, 384], second layer of shape [384, 1]). Additionally, dropout is applied within the layers during training.

Paraphrase Detection For paraphrase detection, sentences s_1 and s_2 are again first encoded into representative embeddings with BERT. In particular, the $[CLS]$ token embeddings are then concatenated before being once again passed into a two-layer MLP with ReLU activation (first layer of shape [1536, 768], second layer of shape [768, 1]). Dropout is applied to the MLP during training.

5 Experiments

5.1 Data

- **1 mil. Wikipedia sentences** - Finetuning BERT using unsupervised SimCSE and DiffCSE (Gao et al., 2022a).
- **Stanford Sentiment Treebank (SST)** - Used downstream for training/evaluating on sentiment analysis (Socher et al., 2013).
- **Quora Dataset** - Used downstream for training/evaluating for paraphrase detection.
- **SemEval STS Dataset (STS)** - Used downstream for training/evaluating for textual similarity (Cer et al., 2017).
- **SNLI** (MacCartney and Manning, 2008) - Finetuning BERT using supervised SimCSE.
- **MultiNLI** (Williams et al., 2018) - Finetuning BERT using supervised SimCSE.

5.2 Evaluation method

5.2.1 Pretraining Evaluation Metrics

For the evaluation of our pretraining strategies we used:

- **Alignment** of positive embedding pairs (lower is better), which can be calculated via:
$$\mathbb{E}_{(x,x^+)\sim\ p_{pos}}||f(x)-f(x^+)||^2 \tag{7}$$
- **Uniformity** of all embeddings (higher is better) which can be calculated via:
$$\log \mathbb{E}_{i.i.dx,y\sim\ p_{data}} e^{-2||f(x)-f(y)||^2} \tag{8}$$
- **Frozen Weights for Downstream Training**, which only performs gradient steps on classifier weights. This way we assessed the quality of the sentence embeddings passed to the classifiers much faster than by finetuning the entire model.

5.2.2 Classifier Evaluation Metrics

Evaluation of the downstream multitask classifier can be divided into each particular NLP task:

- **Sentiment Classification** Average accuracy between predicted and true sentiment values.
- **Paraphrase Detection** Average accuracy between predicted and true paraphrase scores.
- **Similarity** Pearson correlation coefficient between predicted and true similarity scores.

5.3 Experimental details

5.3.1 Datascaling

It was often infeasible or unnecessary to train on 100% of the data, so we decided to implement a sliding data-scale argument to our training algorithm. Particularly, datascaling "sliced" the data training data (i.e a datascale of 0.1 would take the first 10% of the dataset) instead of sampling in order to keep training runs deterministic.

Pretraining Strategies
For each pretraining strategy, the original datasets were provisioned, but would only retain $x\%$ of data for training.

BERT	Unsupervised SimCSE	Supervised SimCSE	DiffCSE
1	0.2	1	1

Final datascaling weights chosen across pretraining strategies.

Particularly, we found that unsupervised SimCSE performed somewhat better with a smaller datascale.

Classifier Finetuning
For the classification tasks, there was no data-scaling used during training.

5.3.2 Batch Size

Pretraining Strategies
An important aspect of consideration was the batch size given for pretraining. For contrastive learning, the batch size can actually play a large role in the training quality. Referring back to 1 and 2, it should be noted that N is a mini-batch of pairs. Therefore a small batch size could result in a noisy loss, and inaccurate gradient updates. On the hand, a big enough batch size would cause out-of-memory (OOM) errors with our available arsenal of GPUs. As, such the following batch sizes were selected:

Unsupervised SimCSE	Supervised SimCSE	DiffCSE
64	32	64

Final batch sizes chosen across pretraining strategies.

Classifier Finetuning
Unlike the pretraining strategies, the batch size for each classification task did not directly affect the final loss.

STS	SST	Paraphrase
8	8	32

Final batch sizes chosen for classification tasks.

A point of distinction within our strategy was to increase the paraphrase batch size to 32, given it's abundance of training data. This increased batch size subsequently doubled our training speeds in practice.

5.3.3 Misc. Hyperparameters

For the rest of the hyperparameters, we will provide a table of chosen values, alongside brief explanation of the seen effect.

Pretraining Strategies

Hyperparameter	BERT	Unsupervised SimCSE	Supervised SimCSE	DiffCSE
Learning Rate	10^{-5}	10^{-5}	10^{-5}	10^{-6}
Weight Decay	0.1	0.1	0.1	0.1
Dropout	0.1	0.1	0.1	0.1
τ	NA	0.05	0.05	0.05
λ	NA	NA	NA	0.005

Final miscellaneous hyperparameters chosen across pretraining strategies.

Learning Rate was found to be optimal around $10^{-}5$, with larger learning rates like $10^{-}3$ causing training to oscillate. $10^{-}6$ was also tested, but found to be slightly worse across the board.

Weight Decay was included to combat against potential over-fitting, but was found to not have a noticeable effect on the final output. In the end, we decided to include it in case of unseen benefits.

Dropout was also included as a method to improve robustness and was kept consistent at 0.1.

τ (from equation 1) remained consistent across runs at 0.05.

λ (from equation 7), was set at the original paper's suggestion of 0.005.

Classifier Finetuning

Hyperparameter	BERT	Unsupervised SimCSE	Supervised SimCSE	DiffCSE
Learning Rate	10^{-6} & 10^{-3}	10^{-6} & 10^{-3}	10^{-6} & 10^{-3}	10^{-6} & 10^{-3}
Weight Decay	0.1	0.1	0.1	0.1
Dropout	0.1	0.1	0.1	0.1

Final miscallaneous hyperparameters chosen across pretraining strategies.

We chose different starting learning rates for the pretrained BERT model 10^{-6}, and the classification heads 10^{-3}. This approach accelerated the classifier's training process while simultaneously protecting against forgetting previously acquired knowledge from the pretraining phase. We also tested a learning rate of 10^{-5}, however we noticed that the results were slightly worse across each task.

5.3.4 Performance Improvements

Max Sequence Length for Contrastive Strategies Given the importance of batch size within contrastive learning, achieving a large batch size of 32 or even 64 required some augmentation of the training data. Specifically, forcing all input sentences to conform to a max sequence length would ensure a given batch would fit into GPU VRAM.

Unsupervised SimCSE	Supervised SimCSE	DiffCSE
64	100	64

Final sequence length chosen across contrastive learning strategies.

A key difference in our methodology for supervised SimCSE involves the handling of sentences exceeding 100 tokens. Unlike the unsupervised version, where sentences were truncated, we opted to exclude sentences beyond this threshold in the supervised model. This decision was made to preserve semantic integrity, as truncating could lead to the loss of critical meaning. This modification impacted only 0.2% of the pairs.

5.3.5 Round Robin Classification Training

While finetuning the classifier, we found that the method in which we train each downstream task affected the final output quality. As such, we decided upon a "round-robin" approach in which each task is trained for only 1 epoch, before being rotated to the next task. This is in direct contrast to training each subtask to compeltion before switching.

5.3.6 Training Environment

For running our models, we opted to using Google's colab environment as opposed to the GCP platform. We found that there a couple instances in which GCP VMs would unexpectedly fail, so as such colab proved to be the more reliable environment for training.

Within colab, we used a mix between the T4 and V100 GPUs, with later experiments leaning more heavily upon the V100 in order to speed up training times.

5.3.7 Training Time

The training times on a single V100 GPU are as follows:
Unsupervised SimCSE: 1 hour per epoch.
Supervised SimCSE: 45 minutes per epoch.
DiffCSE: 1 hour and 15 minutes per epoch.
Multitask Classifier: 25 minutes per epoch.

5.4 Results

5.4.1 Alignment & Uniformity across Pretraining Strategies

Model	Epochs	Alignment	Uniformity
Unsupervised SimCSE 20% data	8	88.747	-132.213
Unsup. SimCSE 100% data	1	95.421	-137.453
Sup. SimCSE Author's dataset	3	152.70	-3.505
Sup. SimCSE Augmented dataset	3	138.102	-3.402
DiffCSE	1	116.095	-153.726

Alignment & Uniformity results per pretraining strategy.

Lower alignment on the augmented dataset for Sup. SimCSE Augmented dataset is expected because the L2 distance of the augmented entailment pairs should be smaller as they are the same sentences. In addition, for the unsupervised SimCSE, we notice that we can reach a lower alignment and similar uniformity using less data, but training for more epochs.

5.4.2 Classification Results

Pretraining Strategies

All models below were trained using 10 epochs.

Model	SST accuracy	Paraphrase accuracy	STS correlation
Base BERT	0.502	0.779	0.842
Unsupervised SimCSE 20% data	0.490	0.838	0.869
Unsup. SimCSE 100% data	0.468	0.842	0.866
Sup. SimCSE Author's dataset	0.511	0.832	0.877
Sup. SimCSE Augmented dataset	0.484	0.849	0.878
DiffCSE	0.441	0.839	0.853

Final results on DEV downstream tasks

From the dev results, it appears that the supervised SimCSE using the author's dataset approach appears to give the highest results upon the dev benchmarks. We were surprised to see that our data augmentation approach didn't result in any substantial performance increases to supervised SimCSE. We theorize this may be a symptom of quality vs quantity, in which Supervised SimCSE benefits more from curated, high-qualtiy data over a larger dataset.

We were also surprised to find that DiffCSE performed moderately worse than SimCSE on SST accuracy and STS correlation (while paraphrase accuracy remained relatively consistent). One theory we have is that the chosen STS and SST classifier heads might not have been optimal strategies for DiffCSE embeddings. That may explain why paraphrase remained performant, but SST and STS saw significant decreases.

5.4.3 Test Results

Model	SST accuracy	Paraphrase accuracy	STS correlation
Sup. SimCSE Author's Dataset	0.528	0.832	0.874
Sup. SimCSE Augmented dataset	0.500	0.842	0.874

Final test results on TEST downstream tasks

6 Analysis

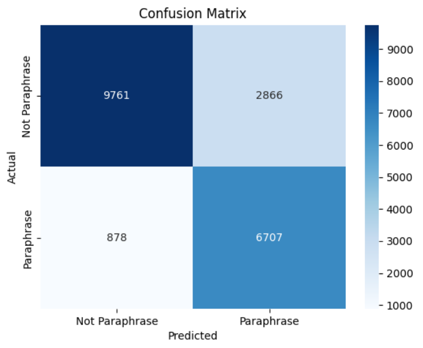
6.1 Similarity Classification Head

Something we found a bit strange was how the final MLP within our similarity classification head operated. We found that a two-layer MLP with shapes [1, 10] and [10, 1] actually performed surprisingly well.

Decomposing a scalar feature into 10 different scalars seems a bit counter-intuitive, but one possible explanation is that MLP is learning a really specific non-linear scaling pattern that properly matches between cosine similarity and the actual similarity threshold.

6.2 Paraphrase false-positive analysis

We observed that, in terms of accuracy, our paraphrase classification task has not been performing at par with leading entries on the leaderboard. Consequently, we embarked on an investigation to understand the potential causes behind this under-performance.



Confusion matrix for the paraphrase task.

By utilizing the confusion matrix, we can see that our false-positive predictions significantly outnumber the false negative ones. Some examples include:

- How do I become great? - How do I become a great doctor?
- How can I lose fat as a teenager? - How can I lose fat as a 15 year old?
- How can i get freebies in india? - Where can I get Freebies in India?

We notice two cases where our model underperforms. The first one is when both sentences share a large number of words with each other, and the second one is when they are semantically very close, but not close enough to be paraphrases of each other.

7 Conclusion

As a result of our experimentation, we found that the author's version of supervised SimCSE to achieve the highest performance on textual similarity, sentiment classifcation, and paraphrase detection. Additionally, we found that for unsupervised SimCSE, we could achieve similar performance results by using less data for more epochs.

Overall, we feel that these results align pretty well with the respective SimCSE and DiffCSE papers, but were surprised that the data augmentation technique for supervised SimCSE did not provide adequate results. Additionally, we were surprised at DiffCSE's lackluster performance on downstream tasks. Future work could look into other augmentation techniques for contrastive learning.

References

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In

Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljačić, Shang-Wen Li, Wen tau Yih, Yoon Kim, and James Glass. 2022. Diffcse: Difference-based contrastive learning for sentence embeddings.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, page 55–65, Online. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022a. Simcse: Simple contrastive learning of sentence embeddings.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022b. Supervised simcse - author's dataset.

Mei Kobayashi and Koichi Takeda. 2000. Information retrieval on the web. *ACM computing surveys (CSUR)*, 32(2):144–173.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528, Manchester, UK. Coling 2008 Organizing Committee.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.