Reward Search: Deep Metric Learning for Information Retrieval with Multiple Objectives

Marcus Warren
Queen's University
18mgw1@queensu.ca

Adam Clarke*
Queen's University
21asc6@queensu.ca

Basil Jancso-Szabo*
Queen's University
18bjjs@queensu.ca

Brett Hobbs
Queen's University
20bhh3@queensu.ca

Alexander Marinkovich
Queen's University
21agm26@queensu.ca

*Equal Contribution

***Abstract*—Information Retrieval (IR) systems are at the core of search engines, data mining and recently Retrieval Augmented Generation (RAG), which is used to provide context to generative AI for knowledge intensive tasks. However, modern deep learning methods used in IR struggle to adapt to objectives that they were not trained for. Existing IR models require fine-tuning to accommodate for downstream tasks. This requires access to large amounts of labeled data and can potentially harm the models ability to generalize. Additionally fine-tuning large models means that their encoded outputs cannot be reused for other systems.**

To solve these problems we propose Reward Search, a novel loss function and IR architecture capable of retrieving information with multiple user defined objectives. We demonstrate the models capability by playing the language game Codenames [Chvátil, 2015] which requires finding text from a knowledge base that satisfies multiple parameters. Reward Search takes 10 minutes to train on mid-range commercial hardware, is capable of handling a dynamic number of inputs and all trainable parameters can be stored in a 50MB file.

I. Introduction

Information Retrieval (IR) is the process of finding and accessing information from a knowledge base that is relevant to a query or set of conditions. In the field of Machine Learning (ML) this task involves encoding data into a fixed size vector/embedding representation which can be compared to other data points using a similarity or distance metric [Wang et al., 2021]. By representing information as points in a latent space, Approximate Nearest Neighbor (ANN) search can be used to efficiently find information relevant to an input query [Malkov and Yashunin, 2018].

In the field of deep learning, achieving optimal IR performance involves training an encoder capable of generating rich text, image, audio or multi-modal embeddings in which their position in a latent space is representative of their semantic meaning related to its retrieval task [Reimers and Gurevych, 2019]. While most modern IR models are trained for zero-shot learning, adapting the behaviour of the model requires fine-tuning. However, this comes with several problems. The training time is relatively long due to a large amount of trainable parameters, the encoder must be retrained to support new behaviour and fine-tuning can lead to the model losing its ability to perform well in general tasks. Instead, we propose the use of a 'search head' which is a relatively small feedforward network trained to map encoder outputs to a new position in the latent space that results in improved performance for downstream tasks. This is accomplished by incorporating ANN search directly into the objective function, and training the model to generate an embedding near areas of higher reward.

The current methodology for training IR models is most often treated as a metric learning problem [Lewis et al., 2021]. In these problems, models such as Siamese Networks are trained to move more similar embeddings closer together and less similar embeddings farther apart [Reimers and Gurevych, 2019]. Modern metric learning tasks focus on either Contrastive or Triplet Loss as the basis for training. Triplet loss has been found to perform better than Contrastive loss in generating learned feature embeddings as it is able to better learn overall structure of the feature space [Wang et al., 2021]. Triplet loss uses a positive, negative and an anchor embedding, and works to minimize the similarity between the positive and anchor embeddings while maximizing the dissimilarity between the positive and negative values [Reimers and Gurevych, 2019]. For search tasks however, we treat the model output as the anchor and we utilize clustering methods as seen in Ranked List Loss [Wang et al., 2021] to group embeddings into positive and negative points relative to the search task. We refer to the model output as the perspective point, and clustered groups of embeddings are considered to be either positive or negative relative to this perspective.

The Reward Search algorithm is trained to play the language game Codenames [Chvátil, 2015], which is treated as a retrieval task with multiple objectives. While Codenames is played with only words, we test the models performance on sentence long texts as well.

A. Codenames

Codenames is a word game where 25 words are selected at random, 9 words belong to the red team, 9 belong to the blue team, 6 words are neutral and the final word is an assassin word. Each team has two roles, the Spymaster and the Operative. The Spymaster is able to see which words belong to which word class, while the Operative cannot. The Spymasters goal is to give a one word 'hint' to the Operative, which the Operative will then use to pick the words on the board that they believe belong to their team. The words are chosen sequentially and the Operative picks as many words as possible until they select a word that does not belong to their team. Not

all incorrect guesses have the same weighting. If a word from the opponents team is picked, then their opponent gets a point, and if the assassin word is picked the current players team automatically losses. Thus, the objective of the Spymaster is to pick a word that will result in the most amount of target words being guessed while also having the incorrect word selection be a neutral word rather than a negative or assassin word. Since the players turn ends as soon as a non-target word is selected, the order of words after the first guessed non-target word does not matter and is not factored into the reward function.

Using Codenames as the foundation for our project allows for easy access to data, clear metrics for evaluating model performance, and demonstration of the algorithms ability to handle complex objectives. We were inspired by Domain Randomization, a technique often used for training autonomous robots in simulated environments [Tobin et al.,]. By randomizing all possible board states, the model is less likely to overfit to biased data and is forced to learn the underlying semantic relationships between its inputs.

B. Motivation

The project is motivated by a perceived limitation of modern deep learning-based search systems, which can only accept a single query as an input. This is believed to be a significant limitation of IR models as they are only trained to capture the similarity between certain types of inputs. For example, a question/answer model would not perform as efficiently as a question/question model.

Unlike standard search related tasks such as question/answer or text to image retrieval, Codenames requires multiple inputs. Additionally the number of inputs must be able to account for multiple game states. Using a string representation of all words is a potential solution to this problem. However, this solution does not scale to longer texts and requires extensive fine tuning of the encoder backbone. Instead the model relies entirely on clustering techniques inspired by Ranked List Loss [Wang et al., 2021] to support multiple words of the same class. The model takes 4 embeddings as inputs, where each input is the normalized and mean pooled value of all embeddings of their respective word class. This allows for inference over a dynamic range of values, with a relatively small number of parameters. Additionally the encoder does not need to be fine-tuned which means that it can be reused for other tasks and does not need to be loaded into memory during training.

C. Problem Definition

The aim of this paper is to demonstrate the Reward Search architecture's ability to retrieve information that fulfils multiple criteria, while also being able to have that criteria being capable of adjustment. Unlike existing bots that use embeddings to play Codenames [Jaramillo et al., 2020] ours will not rely on the construction of a weighted search graph to find the optimal output. This method does not scale to longer texts and is not capable of parallelization. Reward Search relies on a single forward pass and is capable of being processed in batches, additionally it accepts any text below 512 tokens. While the vector search algorithm does rely on graph based search methods, it does not need to be rebuilt after every game as it is only based on nearest neighbor rules.

II. Related Work

Deep Metric Learning seeks to learn embeddings that can capture semantic similarity and dissimilarity information among data points [Schroff et al., 2015], [Wang et al., 2021], [Weinberger et al., 2005]. While originally built for image related tasks, it has seen recent use in text based and multi-modal applications [Reimers and Gurevych, 2019]. The created vector space allows for the use of metrics such as the Euclidean distance and cosine similarity to be used to compare how similar two data-points are to each other [Wang et al., 2021], [Weinberger et al., 2005].

Contrastive loss is outperformed by triplet loss [Schroff et al., 2015], where an anchor point is compared against one positive example and one negative sample. Triplet loss aims to pull the anchor point closer to the positive point than the negative point by a fixed margin [Schroff et al., 2015]. This forces embeddings from a given class to live on the surface of a manifold, which maintains intraclass variances alongside discern-ability from other classes [Schroff et al., 2015]. This is ideal for learning embeddings with useful metrics, where anchors are moved closer to similar data points and further from dissimilar data points [Wang et al., 2021], [Sohn, 2016], [Song et al., 2017], [Movshovitz-Attias et al., 2017], [Schroff et al., 2015].

Ranked List Loss extends this notion by exploiting all negative and all positive examples in a mini-batch [Wang et al., 2021]. Further, instead of pulling elements of a class onto an arbitrary manifold, the elements are pulled onto a hypersphere [Wang et al., 2021]. This maintains intraclass variances, improving generalization performance [Wang et al., 2021].

A. Language Representation Models

Word embeddings apply Deep Metric Learning to the semantics of words, representing words as continuous vectors in a low-dimensional space. This enables further analysis of the words and enables the simple comparison of words using metrics such as cosine similarity [Mikolov et al., 2013], [Pennington et al., 2014].

This work has been extended to embedding sequences of words using a variety of methods [Arora et al., 2017], [Cer et al., 2018] but most notably using Siamese Networks [Reimers and Gurevych, 2019]. By using the same network to generate positive and negative pairs of texts and then using an objective function to increase the distance/dissimilarity has allowed modern networks to capture the relationships between longer bodies of texts.

For SBERT and 'all-mpnet-base-v2' [Hugging Face, 2024], during training the outputs of the two Siamese Networks are normalized before computing their dot product [Reimers and Gurevych, 2019]. This product is the cosine similarity between the two outputs. Triplet loss aims to minimize the

cosine difference between two points relative to the anchor, separated by a margin [Reimers and Gurevych, 2019]. Thus, we can assume all learnable parameters in the loss function are magnitude invariant.

This is due to all embeddings existing along the surface of a manifold with a constant learnable curvature [McInnes et al., 2020]. Using relative differences between positive and negative points allows for the model to learn from a local metric space [McInnes et al., 2020], [Reimers and Gurevych, 2019]. Thus the relationship between two respective features is relative to a perspective point. For example the words 'sick' and 'skateboarding' are more related relative to the word 'coolness' than they are to the word 'fever'.

B. Codenames Bots

There are few published Codenames bots, so they will all be listed.

Andrew Kim et. al. created two bots to play the game: a Codemaster and a Guesser [Kim et al., 2019]. The codewords were embedded using both Glove [Pennington et al., 2014] and Word2Vec [Jaramillo et al., 2020], and a concatenation of both [Mikolov et al., 2013]. The Codemaster selected a clue where the clue was further from the non-target words than from the worst target word, and the distance between the cue and the worst target word is less than some threshold [Kim et al., 2019]. The guesser would then repeatedly guess the closest word to the clue until a mistake was made.

Catalina M. Jaramillo et. al. improved on the results of Kim et. al. using GPT-2 [Radford et al., 2019] word embeddings [Jaramillo et al., 2020]. They further demonstrated that performance could be improved if a weighting system was introduced, penalizing the selection of the other team's target words or the assassin more than the neutral words.

Divya Koyyalagunta et. al. use BabelNet [Navigli and Ponzetto, 2012], a word representation graph, to generate and guess clues [Koyyalagunta et al., 2021]. They are able to produce similar results to Catalina et. al. while allowing for the interpretability of the model. This is because BabelNet provides the relationships between objects in a human understandable form, such as

$$\text{moon} \xrightarrow{\text{references in description}} \text{planet} \xrightarrow{\text{is a}} \text{celestial body}$$

which enables direct interpretation [Navigli and Ponzetto, 2012], [Koyyalagunta et al., 2021].

C. Dataset

The project makes use of two separate datasets for the word only task, the first for the game board and the second for the models vocabulary. The game board dataset is composed of 400 words, used in a separate Codenames project [Wiman, 2024] and is used to create all potential game boards that the model is trained on. The vocabulary dataset represents all possible words that the model can choose and is composed of all words found in the Merriam Webster dictionary, representing over 63,000 potential choices. All words that exist in the game board dataset are removed from the vocabulary dataset, this is done to prevent the model from choosing words that are exactly the same as the ones on the board. All words are then encoded into embeddings using the 'all-mpnet-base-v2' Sentence Transformer, to allow for faster inference.

In the long text trials both the game board and vocabulary dataset use a sampled collection of titles from the Cornell Newsroom dataset [Grusky et al., 2018]. The dataset was split into a board dataset of 800 article titles and a vocab dataset of 80,000 titles. No article titles appear in both the vocab and game board datasets.

For both the word and sentence trials, 100,000 randomly selected groups of 25 pieces of text were chosen from their respective game datasets, and were randomly assigned a positive, negative, neutral or assassin label.

III. Model

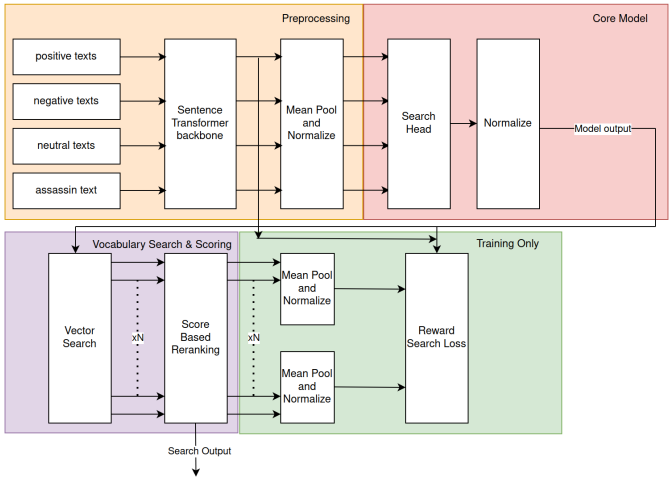

Fig. 1. Reward Search training and inference architecture

The architecture of the model is shown in Figure 1 and is divided into 4 main sections. The preprocessing of text is done before training and the embeddings are stored in either a database or external file, allowing for fast training times. The core model is a single feedforward network with 2 hidden layers and contains all trainable parameters for the model. The vocabulary search uses the HNSW vector search algorithm to search through all potential output texts and find the closest number of texts determined by a search window. The texts are then scored and ranked based on the reward function, which was based on the game Codenames. All found texts are then split in half, where the first 50% are the highest scoring texts and the remaining 50% are the lowest scoring. Both half's of the found text embeddings are then mean pooled and normalized and used in the loss function. The loss function also requires all text embeddings that were used as the models inputs. During inference this step is not done and instead only the highest scoring search output is returned.

A. Loss Function

The loss function takes heavy inspiration from triplet loss with a few key changes. Unlike standard triplet loss which uses positive and negative pairs compared to a ground truth, Reward Search works in reverse. Comparing a perspective point to both positive and negative ground truths. The formula for triplet loss is given as:

$$\max(D(x_a, x_p) - D(x_a, x_n) + \alpha, 0) \tag{1}$$

where x_a, x_p and x_n is the anchor, positive and negative embeddings, D is the distance/cosine similarity function and α is the margin. However, to work for search tasks several modifications have to be made.

The anchor point is removed and replaced with x_ϕ which is the perspective point. As x_ϕ is not a ground truth value, the order of the distance measures must also be changed.

$$\max(D(x_\phi, x_n) - D(x_\phi, x_p) + \alpha, 0) \tag{2}$$

In this modified version of triplet loss, the model is penalized for having a perspective closer to unwanted values and rewarded for being closer to positive values.

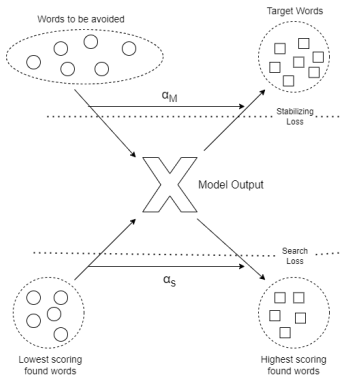

Fig. 2. graphical depiction of Reward Search Loss

Figure 2 depicts a graphical representation of how the loss is applied to the model output. Equation 2 is applied twice, both on the models inputs and the found search embeddings. Only comparing the model output to its highest and lowest scoring found embeddings is not enough for the model to learn to play the game. Instead a stabilization portion must be added. How this might be applied to models with only a single input vector is still not known but it could potentially be solved by using ANN search around the resulting input. This however, limits Reward Search to require at least 4 comparison points.

Additionally, it was discovered via experimentation that using single text embeddings as ground truths was not enough for the model to learn to complete the objective and thus, clusters of values are used instead. While it is possible to use non clustered embeddings as inputs, they are less effective. Whereas using single embeddings for the search loss results in noticeably worse results. The decision to use exactly 50% of all found embeddings in a search window is also deliberate as it was discovered to lead to the best performance.

B. Experiment Setup

The model was trained for 10 epochs with a batch size of 500, using a dataset of 100,000 collections of randomized text and word embeddings. All training was done using a GTX 1660ti mobile GPU, taking about 10 minutes to train in total.

C. Evaluation Methods

The model is evaluated based on the results of both its search and model output. The search output is the highest scoring embedding found after the ANN search and re-ranking. The model output is the perspective point used in ANN search. While the search output is the final result of the inference pipeline, the output of the core model is important to determine whether the model is able to learn the objective of the game without being provided the scoring method directly. It also provides insight towards how the perspective point behaves relative to the output of the search.

Both the search and model outputs are evaluated using the same metrics. How many target words does it select before making a mistake and the rate at which it selects neutral, negative and assassin words as the first non-target guess. While the primary goal is to select as many target words as possible, the model must also have a neutral selection rate greater than both the assassin or negative rate, which can be adjustable based on the reward.

IV. Results and Discussion

Figures 3 & 4 demonstrate the behaviour of the model output as the reward for selecting neutral texts over negative & assassin texts increase. The assassin rates are removed due to their low values.

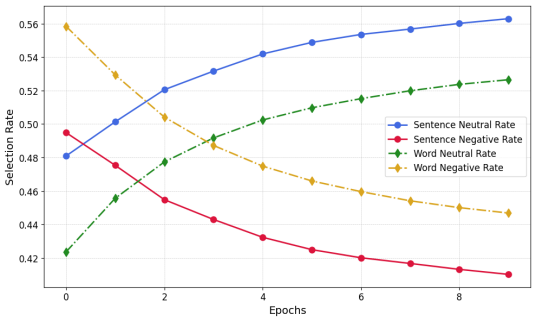

Fig. 3. Change in neutral and negative selection rates by the model output with a neutral reward of 2 for both sentence and word trials

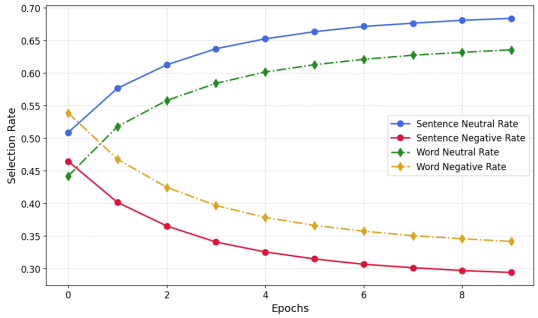

Fig. 4. Change in neutral and negative selection rates by the model output with a neutral reward of 5 for both sentence and word trials

Figures 3 & 4 both demonstrate the models ability to learn the objective of the task without having it applied to the loss function directly. By increasing the neutral reward both sentence and word models selection rates for neutral increase and negative rates decrease. It should also be noted that models trained on longer texts appear to have improved selection rates that converge towards a higher neutral rates faster than the word model. The text model was trained on a larger game board and vocabulary dataset alongside an encoder backbone that was trained specifically for text. Thus, these results can not confirm that the semantic 'richness' of long text compared to words leads to an improvement of satisfying multiple objectives. However, it does provide some evidence towards this.

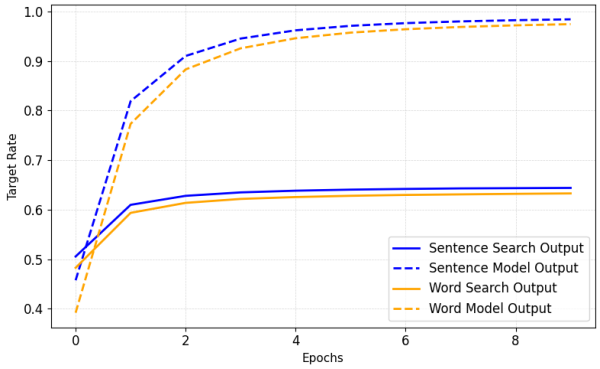

Fig. 5. Target selection rate for both word and sentence embeddings

Figure 5 is a depiction of the models target selection rates over the course of training. The model output has a considerably larger selection rate than the search output for both long text and words. However, it can be seen in tables I & III below, that the perspective point is less capable of completing the secondary objective of non-target word selection. Furthermore, during experimentation this result has been consistent with every combination of hyper-parameters given, including the changing of both the search and margin values used in the loss function. As for overall performance, the model appears to perform far better than theoretically possible for a human who on average can typically only select about 22-39% of the target words per turn. There are no benchmarks for the game available online, however, during testing it was found to be near impossible to beat.

neg w	neut w	assas w	neg rate	neut rate	assas rate	targ selection
0.0	1.0	−10.0	0.52	0.45	0.03	8.77
0.0	2.0	−10.0	0.45	0.52	0.03	8.75
0.0	5.0	−10.0	0.34	0.63	0.02	8.69
0.0	2.0	−100.0	0.45	0.53	0.03	8.75
3.0	0.0	−10.0	0.22	0.75	0.03	8.84

TABLE I
PERFORMANCE OF THE WORD MODEL OUTPUT WITH VARIATION OF THE SEARCH REWARD

neg w	neut w	assas w	neg rate	neut rate	assas rate	targ selection
0.0	1.0	−10.0	0.25	0.75	0.00	5.70
0.0	2.0	−10.0	0.06	0.94	0.00	5.44
0.0	5.0	−10.0	0.00	1.00	0.00	5.27
0.0	2.0	−100.0	0.06	0.94	0.00	5.45
3.0	0.0	−10.0	1.00	0.00	0.00	5.53

TABLE II
PERFORMANCE OF THE WORD SEARCH OUTPUT BY VARYING THE SEARCH REWARD

neg w	neut w	assas w	neg rate	neut rate	assas rate	targ selection
0.0	1.0	−10.0	0.26	0.74	0.00	5.80
0.0	2.0	−10.0	0.07	0.93	0.00	5.55
0.0	5.0	−10.0	0.00	1.00	0.00	5.36
0.0	2.0	−100.0	0.07	0.93	0.00	5.55
3.0	0.0	−10.0	1.00	0.00	0.00	5.66

TABLE III
PERFORMANCE OF THE LONG TEXT SEARCH OUTPUT BY VARYING THE SEARCH REWARD

Figures 3 and 4 illustrate that the model output of the long text model demonstrates a superior selection rate compared to the word model. However, in I & III the resulting search output of the long text model performs worse at selecting non-target words. While the target selection rate appears consistent with the model output the secondary objective does not.

V. Conclusion

Reward Search has demonstrated its ability to retrieve information with adaptable objectives. The algorithm is fast to train with a small parameter count. The model output demonstrates signs of learning the objective function at a game of Codenames without it being applied to the loss function directly and is capable of playing the game more effectively than a person.

VI. Future Work

The models behaviour needs to be tested on different encoders, including image encoders. Additionally the model should be tested on different scoring objectives not based on the game of Codenames. Single topic retrieval is the logical next step for the utility of the algorithm. Being able to retrieve information about a topic that satisfies multiple objectives

would allow for more controllable ML based search engines as well as RAG.

Testing the model on embeddings of varying sizes should also be looked into. Due to the structure of the latent space being roughly geodesic [McInnes et al., 2020] decreasing the size of the embeddings will lead to an increased surface area/volume of the space (consequence of high dimensional geometry), and it would be interesting to see if there is a relationship.

Additionally training the model on embeddings generated by encoders pretrained using different activation functions could lead to further insights into the structure of the latent space. If an encoder is trained using hyperbolic tangent instead of GELU, would the model perform the same?

VII. Limitations

Reward Search is a new method for IR tasks and as such, the full scope of its abilities are currently unknown. How does the size of the embeddings effect performance? While it works on models trained via metric learning methods are there other situations where it can be used? The objective of Codenames is based on similarity metrics, but can it be used with other non-similarity based objectives? Is it possible to retrieve information based on a topic that also possesses a required sentiment. Modifications to the core model design will likely need to be made to fulfil these tasks.

The current framework is theorized to work with any embedding, so long as that embedding can be modelled to exist along the surface of a manifold with a relatively consistent, learnable curvature. The position of the embedding must also be representative of its meaning and thus Reward Search will likely be less efficient for encoders that have not been trained on large amounts of general information. While we have shown that training a model to search for values within a latent space is possible, we do not know the exact limitations for what values are possible to be searched for. More research is required.

VIII. Data Availability

All code used to train and test the model is available at https://github.com/marcus-wrrn/RewardSearch.

References

[Arora et al., 2017] Arora, S., Liang, Y., and Ma, T. (2017). A Simple but Tough-to-Beat Baseline for Sentence Embeddings. Accepted: 2021-10-08T19:51:06Z.

[Cer et al., 2018] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., and Kurzweil, R. (2018). Universal Sentence Encoder. arXiv:1803.11175 [cs].

[Chvátil, 2015] Chvátil, V. (2015). Codenames.

[Grusky et al., 2018] Grusky, M., Naaman, M., and Artzi, Y. (2018). Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

[Hugging Face, 2024] Hugging Face (2024). all-mpnet-base-v2.

[Jaramillo et al., 2020] Jaramillo, C., Charity, M., Canaan, R., and Togelius, J. (2020). Word Autobots: Using Transformers for Word Association in the Game Codenames. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 16(1):231–237. Number: 1.

[Kim et al., 2019] Kim, A., Ruzmaykin, M., Truong, A., and Summerville, A. (2019). Cooperation and Codenames: Understanding Natural Language Processing via Codenames. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 15(1):160–166. Number: 1.

[Koyyalagunta et al., 2021] Koyyalagunta, D., Sun, A., Draelos, R. L., and Rudin, C. (2021). Playing Codenames with Language Graphs and Word Embeddings. *Journal of Artificial Intelligence Research*, 71:319–346.

[Lewis et al., 2021] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2021). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401 [cs].

[Malkov and Yashunin, 2018] Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. arXiv:1603.09320 [cs].

[McInnes et al., 2020] McInnes, L., Healy, J., and Melville, J. (2020). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv:1802.03426 [cs, stat].

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs].

[Movshovitz-Attias et al., 2017] Movshovitz-Attias, Y., Toshev, A., Leung, T. K., Ioffe, S., and Singh, S. (2017). No Fuss Distance Metric Learning using Proxies. arXiv:1703.07464 [cs].

[Navigli and Ponzetto, 2012] Navigli, R. and Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

[Reimers and Gurevych, 2019] Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv:1908.10084 [cs].

[Schroff et al., 2015] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823. arXiv:1503.03832 [cs].

[Sohn, 2016] Sohn, K. (2016). Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

[Song et al., 2017] Song, H. O., Jegelka, S., Rathod, V., and Murphy, K. (2017). Deep Metric Learning via Facility Location. arXiv:1612.01213 [cs].

[Tobin et al.,] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world.

[Wang et al., 2021] Wang, X., Hua, Y., Kodirov, E., and Robertson, N. M. (2021). Ranked List Loss for Deep Metric Learning. arXiv:1903.03238 [cs].

[Weinberger et al., 2005] Weinberger, K. Q., Blitzer, J., and Saul, L. (2005). Distance Metric Learning for Large Margin Nearest Neighbor Classification. In *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

[Wiman, 2024] Wiman, M. (2024). Codenames. original-date: 2016-11-27T10:09:45Z.