Retrieving Game States with Moment Vectors

Zeping Zhan, Adam M. Smith
Design Reasoning Laboratory
Department of Computational Media
UC Santa Cruz
{zzha50,amsmith}@ucsc.edu

Abstract

Game scholars need to find moments in games that advance their arguments, and artificial intelligence algorithms need to recall states that are most promising for exploration. This paper considers the problem of engineering a representation for game states that is suitable for retrieval in the vector space model. Retrieving moments from gameplay traces for two popular Super Nintendo Entertainment System games, we evaluate several different representations including one derived from a deep embedding of screenshot pixels based on a supervised memory prediction task. The results suggest compact moment vectors may be a promising representation for building future systems that intend to build higher level knowledge about games.

Introduction

In videogames and other interactive media, the space of possible states of a system can be vast. In this paper, we seek to engineer a compact and semantically rich representation of a game's state, preferably one derived from just observations (e.g. screenshots rather than hidden memory state). Looking to natural language processing (NLP) for inspiration, word vectors are compact numerical representations of words that implicitly capture semantic knowledge (Turian, Ratinov, and Bengio 2010). By operating on word vectors rather than discrete words, new NLP systems can reason over texts (sequences of words) with very large vocabularies without the need to learn each word's meaning from scratch. This paper introduces the concept of *moment vectors* for abstracting interactive system states. By embedding observations of a game into moment vectors, we hope to enable future systems to quickly build higher level knowledge.

Such a representation for game moments could unlock a new domain of content-based information retrieval. Given a screenshot from a known game, we might ask a system to find which moments in an indexed sequence of states (such as from recordings of expert play) are most relevant. We might even ask it to synthesize a new complete state which is as faithful as possible to the given query. This capability could empower game scholars, such as those targeted by the Game and Interactive Software Scholarship Toolkit (Kaltman et al. 2017) to search for specific moments contained within culturally impactful games. It could also aid algorithms for automatic exploration, such as Rapidly-Exploring Random Trees (LaValle 1998), in tasks such as mapping spaces of play (Osborn, Summerville, and Meteas 2017).

Beyond retrieval, moment vectors could be used as low-level representations in systems designed for other tasks. They might help answering know-what questions such as "What is the player's location in space or overall narrative progress?" or know-how questions such as "Which action should I take next to make progress towards the given goal?"

This paper offers a first step in engineering a suitable moment vector representation. As a domain of focus, we examine moments in Super Nintendo Entertainment System (SNES) games. The SNES platform was chosen to balance societal relevance, platform complexity, variety of game types, and availability of pre-existing expert gameplay traces. Inspired by deep visual search models such as ProductNet (Bell and Bala 2015), we consider vectors resulting from training on an auxiliary cross-modality prediction task. Across multiple games and multiple notions of relevance, we evaluate the retrieval ability for different vector representations using the mean Average Precision (mAP) metric.

Moment Retrieval Setting

We approach the retrieval problem using the *vector space model* (Manning, Raghavan, and Schütze 2008, Chap. 6). In this model, queries and documents (moments to be retrieved) are represented by vectors in a moderately high-dimensional space (hundreds of dimensions). The relevance of a document to a query is modeled by the *cosine similarity* (dot product of normalized vectors) between their vector representations.

Corpora

Our initial experiments focus on retrieving moments from pre-recorded interaction traces. On websites such as TASvideos,[1] community members collect tool-assisted speedruns (TAS) represented as game controller input sequences. When played back in a specially prepared game platform emulator such as BizHawk,[2] these traces often

[1] http://tasvideos.org/
[2] http://tasvideos.org/Bizhawk.html

demonstrate spectacular feats of virtuosic play such as completing difficult games within just a few minutes. While many speedruns intentionally skip past much of a game's content by exploiting precise timing and glitches, certain traces aim to show off all of the game's content.

For Super Metroid (a platformer game which involves backtracking through previously visited parts of the world after collecting special items), we use a specific[3] "100%" trace that expertly completes the game without mistakes and restarts. For Super Mario World (a linear platformer without mandatory backtracking), we recorded our own trace that demonstrates three of the game's nine worlds with less-than-expert play (including multiple attempts at certain difficult levels). Both traces are approximately one hour in length.

We associate moments in these games with individual display frames. SNES games display 60 frames per second. To keep data sizes manageable for our initial experiments, we consider only one moment per second (every 60th frame in the sequence). For each moment, we store a lossless representation of the game's display (a $224 \times 256 \times 3$ RGB color image with one byte per color channel) as well as the entire state of the game and game platform as modeled by the emulator (approximately 3 megabytes of data capturing processor registers, main memory, video memory, audio memory, and other subsystem state).

For popular SNES games, the game hacking (modding) community has accumulated extensive notes on the precise meaning of various locations in memory (such as for Super Mario World[4]). As these notes are only available for a subset of SNES games and not readily available for games on other platforms, we will only make use of them indirectly: in noting which regions of memory are often used to store player-relevant game state information (rather than for temporary buffers used in operations like decompression which are not directly visible from the player experience).

Relevance

Retrieval problems require a certain notion of relevance to be formalized before a system can be evaluated. In these initial experiments, we consider a hypothetical video search task: given a screenshot, which points in a pre-recorded sequence of moments are similar?

In one (temporal) formulation of relevance, we assume that moments that are near one another in time are to be considered relevant. Given a query frame from time t in the sequence, the boolean relevance of moment i in the same sequence will be judged by a threshold k on the absolute time difference: $|t - i| \leq k$.

This is an imperfect formulation of the retrieval task for our corpora as, either by mandatory backtracking or restarts from mistakes in play, two moments with a large time difference may feel quite similar to one another. Nevertheless, it requires representations to encode a sense of progress within a given level.

We consider three different values of k so as to evaluate the utility of the representations on different time scales. Our $k = 2$ scenario demands that the retrieval system can make very precise judgements. In one hour of play, just $4/3600 = 0.11\%$ of moments will be considered relevant. A retrieval system effective on this timescale could help an artificial intelligence system that was trying to solve for inputs that play along with the appearance of a pre-recorded video found on a video sharing website. Our one-minute or $k = 60$ scenario (3.3% relevant moments in one hour) demands that the retrieval system can generalize beyond the precise look of a certain moment in play and capture longer range structure such as progress through a sequence of levels and worlds or through a narrative. A system effective on this timescale could help scholars identify the context of isolated screenshots even if no indexed corpus comes close to the query. We also consider the approximate geometric midpoint $k = 10$ to understand any time-dependent trends.

In a second formulation of binary relevance (one that accounts for backtracking), we consider spatial thresholds. In Super Mario World, we consider two moments to be relevant to one another if they happen in the same level.[5] In Super Metroid, we check if they happen within the same room tile: roughly one screenfull of player visible terrain.[6]

Evaluation

To evaluate retrieval quality, we consider the *mean Average Precision* (mAP) metric (Manning, Raghavan, and Schütze 2008, Chap. 8) where the averaging happens over several different queries. From each input sequence, we select query moments spaced one minute apart and judge the system's ability to recall moments that are temporally or spatially similar (excluding the query itself). For a given query and ranked result list, the Average Precision metric records the density (ranging from 0.0 to 1.0) of relevant results in the top of the list, averaged over the increasing number of relevant moments retrieved. This metric is maximized when a retrieval system ranks all of the relevant moments at the top of the list and all irrelevant moments below them. We ask our system to rank all of the moments in the corpus (rather than yielding short top-10 list as would be more typical in a web search task).

Moment Vector Representations

In this section, we consider several different moment vector representations, some of which are impractical but serve as useful reference points. We would like our moment vectors to be both compact and derived from only screenshot data (as this is how they are likely to be used in the future), but we acknowledge that, in some sense, the ground truth representation of a moment is the full platform state snapshot.

Mem-3MiB The first representation we consider is the raw state snapshot itself, a vector of byte values in a three million dimensional space. This representation is both impractical (we usually do not have platform snapshots available to use as queries) and unwieldy (3MiB per moment

[3] `http://tasvideos.org/2436M.html`

[4] `http://www.smwcentral.net/?p=map&type=ram`

[5] Identified by "pointer to level's sprite data" (`https://www.smwcentral.net/?p=map&type=ram`)

[6] Identified by "Room's Automap X[/Y] coordinate" (`https://drewseph.zophar.net/Kejardon/RAMMap.txt`)

makes for bloated archives). Having examined RAM map listings for several SNES games, we hypothesize that this representation overemphasizes low-level details that are invisible from the player perspective.

Mem-128KiB The next representation we consider is the 128KiB of memory efficiently accessible by the CPU (sometimes called WRAM[7] and contrasted with PPU/VRAM of similar size). This representation is more compact, but still has many of the problems listed above.

Mem-4KiB Focusing on a region of RAM most often used to store key game state across a wide variety of SNES games (such as the position of characters on the screen, the score, the currently selected weapon, etc.), we next consider just the first 4KiB of WRAM. In the later representations derived using machine learning techniques, we use this segment of RAM as the target for supervised learning tasks.

Mem-PCA256 To compactly represent game state, we would much prefer a smaller vector. To represent the potential fidelity of a 256-dimensional vector representation, we apply Principal Component Analysis (PCA) to find a linear reduction of the 4096-dimensional representation above into this lower dimensional space.

Pix-168KiB So far, every representation has been based on the invisible contents of memory. To support search by screenshot, we next consider using the $224 \times 256 \times 3$ = 168KiB of pixel data as a raw representation of the display. Although this data could be provided by users of a search tool, it is hardly a compact representation (larger than all of WRAM). Even with this much data, a better representation derived from pixels might additionally consider the values of pixels over time as extracted from a short movie.

Pix-PCA256 Next, we consider mapping the raw pixel data down to 256 dimensions using PCA as well. Although a deep image autoencoder (Krizhevsky and Hinton 2011) might do this task better, we retain the use of PCA here as a baseline comparison method.

Just as we know that some bytes of memory have a more direct impact on player experience than others, we expect that some pixels (or pixel patterns) are more salient than others. To derive a representation of screenshots that emphasizes those patterns that are most predictive of the play experience, we setup a supervised machine learning task where the inputs come from screenshot pixels and the target outputs come from the 4KiB segment of memory mentioned above. We train separate models for the separate games.

Pix2Mem-MLP256 As a baseline method for this pixels-to-memory prediction task, we consider a simple Multi-Layer Perceptron (MLP) with a single hidden layer consisting of 256 hidden units. This simple neural network uses the ReLU activation and is trained to optimize a Mean Squared Error (MSE) loss on the byte value regression task.

[7] `https://en.wikibooks.org/wiki/Super_NES_Programming/SNES_memory_map`

Pix2Mem-CNN256 As a more interesting model for the pixels-to-memory prediction task, we consider a deep convolutional neural network including a bottleneck layer with 256 dimensions. This network (for which the precise layer configuration is beyond the scope of this paper) is trained to optimize the average categorical cross-entropy loss on the task of predicting the precise value of each byte of memory (4096 simultaneous 256-way multi-class classification tasks). To generate 256 dimensional moment vectors using this model, we apply only the subset of layers that generate the low dimensional embedding (collectively known as the encoder model). Although the larger predicted memory state may someday be useful for synthesizing new complete game states (from which play might be resumed) given just low-dimensional latent vectors, we have a preference for the compact bottleneck representation.

These representations clearly do not exhaust the space of possibilities; they are designed to give insight into what it is important for moment vectors to represent.

Random Finally, to give a sense of default difficulty for the retrieval problem in the various relevance formulations, we also include mAP results for a trivial retrieval system that uses random scores to sort the results (ignoring the vector representations).

Experiments

For the two moment corpora described above, we constructed several retrieval tasks. Our Super Mario World corpus consisted of 3604 moments of which 90 evenly spaced moments were selected to be used as queries. For Super Metroid, we similarly selected 68 queries from the corpus of 4102 moments. Mean subtraction was applied to the moment vectors of each corpus, and no further pre-processing was performed. For each query, we defined temporal relevance on three different timescales $k \in \{2, 10, 60\}$ and spatial relevance using game-specific memory inspection. Using cosine distance between representation vectors and query vectors to rank moments, we evaluated the effectiveness of each representation strategy using the mean average precision (mAP) metric described above.

Figure 1 captures the results of our experiments. Not represented in these results are the outcomes for informal experiments with several alternative deep convolutional neural network architectures. Although several of these alternatives performed worse than the network reported here, we expect there is ample room for improvement both on the pixels-to-memory prediction task the embedding-for-retrieval retrieval task.

Discussion

First examining the results for the memory-based models (Mem-3MiB, Mem-128KiB, and Mem-4KiB), our hunch that the most significant data in memory is contained within the first 4KiB of WRAM is largely confirmed. Across games and relevance formulations (except for spatial relevance in Super Metroid, Metroid-XY), including a wider view of memory only negatively impacts retrieval performance.

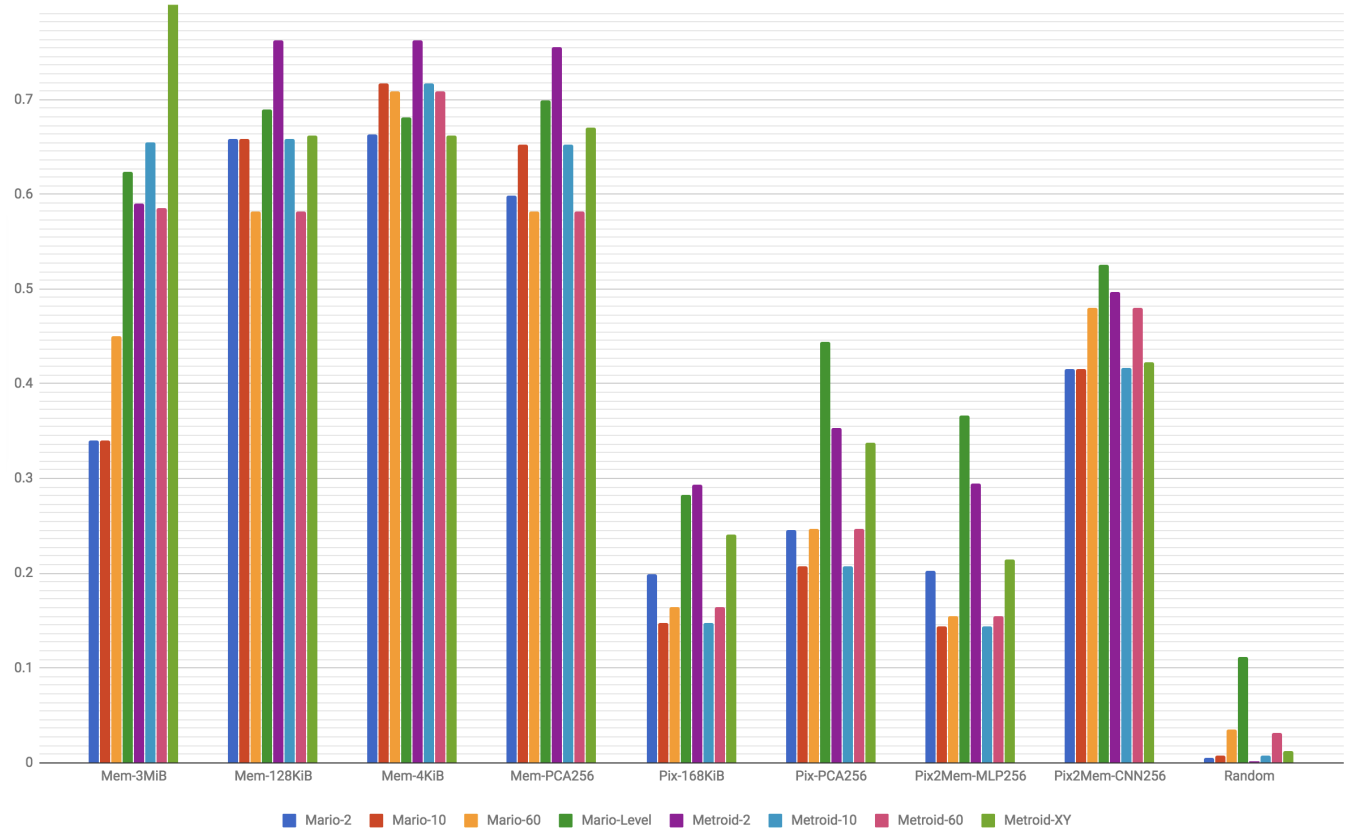

Figure 1: Mean average precision (mAP) scores for retrieval with different moment vector representations (higher is better). Mem* representations are derived from platform memory state while Pix* representations depend only on screenshot pixels. Numbered retrieval tasks use a game-independent temporal notion of relevance while Mario-Level and Metroid-XY use game-specific spatial notions of relevance. Random retrieval performance indicates task difficulty (lower is harder).

Examining the performance of the PCA-reduced view of this important memory window (Mem-PCA256), we see that the dimensionality of the vector can be dramatically reduced (by 16x here) with only modest negative impacts on retrieval performance. In informal experiments, a reduction to just 128 dimensions performed almost as well.

Moving to the pixel-based models, we find large drop in retrieval performance. This is not surprising, however, given that we know that memory contains details not inferable from (individual) screenshots. We conjecture that the raw pixel vector representation (Pix-168KiB) over-represents large patches of background art that do not change with a granularity that is particularly useful for localizing moments. The dimensionality-reduced view of the pixel vector (Pix-PCA256) seems to overcome this slightly by accounting for correlations between pixels.

Looking at the baseline machine learning approach (Pix2Mem-MLP256), we see that extracting the meaningful game state from an image requires more than a simple linear transformation. By contrast, the deep model is able to achieve retrieval performance closer to the memory-based models despite working from only a single frame of pixels.

As suggested above, the notion of temporal relevance used in these experiments is not ideal because it does not consider backtracking or replaying through the same part of a world to be relevant. The fact that this happens in the corpora for the two games considered limits the mAP performance even of systems that could perfectly identify the position of the player's avatar. Systems that perform perfectly on this task, then, are those that are able to accurately encode the time since the input sequence began recording. Given that some elements of memory contain counters (used in implementing state machines for animation), it is possible that some of the apparent retrieval performance in the memory-influenced systems (Mem and Pix2Mem) might be due to a potential learning-to-time effect.

Examining results for the spatial notions of relevance, we see qualitatively similar performance to that for temporal relevance. If the various representations are learning to time, they seem to be learning a usefully local sense of timing.

Results for the Random model suggest there is value in all of the vector representations. In top-10 result lists for a hypothetical screenshot-based moment search engine (as estimated by mAP), the memory-based representations will usually yield 6 relevant results, the deep image analysis representation will usually yield 4, and the other representations

will yield at least one. Meanwhile, for all but the Mario-Level sense of relevance, a random sorting of the results will usually yield zero relevant moments.

In practice, a mature retrieval system would use one or more levels of re-ranking (Yang et al. 2016) to boost the precision of short result lists. In such systems, a generous list of results obtained by cosine similarity search would be re-sorted based on additional criteria not present in the screenshots themselves (such as metadata filters, interactive relevance judgements, or historical click-through data).

Future Work

To better understand the retrieval quality of moment vectors, a small collection of moments could be hand-annotated with their human-perceived relevance to a query image. Acknowledging that users may not have the ideal query images on hand, such a dataset could also record the use of weights on or analogies between multiple query images to indicate relevance to a moment that is not directly represented in any of the query images. To indirectly indicate the state of being in a late-stage boss battle with full health for Super Metroid, the user might subtract two early-game moments for which health difference is the only meaningful change and add this to a moment from the late-stage boss battle with low health.

Towards understanding the ability of moment vectors to support extracting higher level knowledge, the same input sequences used in our experiments (or other speedruns from the TASvideos site) could be used as supervision in a policy learning scenario. Does the moment vector representation allow one to better generalize from limited data? Can training a representation on data from multiple games at a time support knowledge transfer?

The moment vectors in this paper considered the state of the game platform in the instant between display frames. A notion of "moment" perceivable by human players might include what has happened recently (and even what is most likely to happen next). Future experiments should consider reasoning with moment vectors over time, such as with a recurrent neural network, to extract information about a broader notion of moment than the current frame. Alternatively, a screenshot from one second later in the speedrun might be considered as the training target in an image-to-image prediction task that bypasses the need to identify semantically-rich regions of platform memory.

Related Work

Engineering (via machine learning or otherwise) state abstractions has a long history in artificial intelligence. Work such as that by Cobo et al. (2011) demonstrates that human expert behavior can guide the development of useful abstractions that improve the data efficiency of future learning. Abstractions need not be constructed before use, as, Sturtevant et al. (2005) demonstrate building abstractions on-line during the execution of a heuristic search algorithm.

The strategy of engineering vector representations for retrieval by training a deep network on a proxy task (Lin et al. 2015) has recently had major impacts on the field of content-based image retrieval. When retrieval in particular is the intended use of embedded vectors, the learning task can be adjusted to specifically optimize the quality of the embedding for the purposes of judging distances between relevant and irrelevant pairs (Bell and Bala 2015). This is useful when images are being used to retrieve a different type of data. We would like to retrieve very different-looking images that depict semantically-similar moments. Where Bell et al. intend to retrieve household products (such as lamps and couches) despite changes in camera orientation and lighting, we would like to retrieve moments from similar times and locations in a game despite graphical effects such as full-screen flashing or changing user-interface modes.

Ryan et al. (2016) considers the problem of retrieving related games from an embedded vector space. Our work makes two important points of contrast with this. First, we consider content-based retrieval (based on what is to be seen by playing the game) rather than metadata-based retrieval (based on what is written and recorded about the game). Second, we consider indexing corpora of individual moments within a given game while Ryan et al. consider entire games as the objects to be represented by vectors. These perspectives are complementary, and we believe they should be combined in the future. For a system attempting to learn a cross-game perceptual model (going from pixels to compressed state vector for more than one game), having a compact encoding of information around genre, theme, and expected play styles might stabilize training. Likewise, by matching a representation of the contents of a game with a representation of the metadata for popular games, we might be able to project information onto unpopular or as-yet unreleased games for which no external metadata is available.

Conclusions

In this paper, we introduced the problem of engineering moment vector representations. Moment vectors are (ideally) compact vector representations of the most important aspects of the state of a game at a specific point during play (at the granularity of a single frame). By examining different treatments of game memory and visual display information, we were able to define a representation that depends only on displayed pixels while having retrieval performance approaching that of similarly compact models based on access to hidden memory state. We intend for moment vectors to serve the needs of both human game scholars as well as artificial intelligence algorithms that need to quickly recall relevant game states based on visual clues alone.

Acknowledgements

The authors thank Xiaoxuan Zhang for her contribution of the spatial relevance formulation and Misha Holtz for allowing us to test our vector representations in her in-development videogame moment search engine. Our deep convolutional neural network model was developed with Keras (Chollet and others 2015).

References

Bell, S., and Bala, K. 2015. Learning visual similarity for product design with convolutional neural networks. *ACM*

Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 34(98).

Chollet, F., et al. 2015. Keras. `https://github.com/fchollet/keras`.

Cobo, L. C.; Zang, P.; Isbell, Jr., C. L.; and Thomaz, A. L. 2011. Automatic state abstraction from demonstration. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, 1243–1248. AAAI Press.

James, R.; Eric, K.; Timothy, H.; Katherine, I.; Michael, M.; and Noah, W.-F. 2016. GameNet and GameSage: Videogame discovery as design insight. In *DiGRA/FDG'16 Proceedings of the First International Joint Conference of DiGRA and FDG*. Dundee, Scotland: Digital Games Research Association and Society for the Advancement of the Science of Digital Games.

Kaltman, E.; Osborn, J.; Wardrip-Fruin, N.; and Mateas, M. 2017. Getting the gisst: A toolkit for the creation, analysis and reference of game studies resources. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, FDG '17, 16:1–16:10. New York, NY, USA: ACM.

Krizhevsky, A., and Hinton, G. E. 2011. Using very deep autoencoders for content-based image retrieval. In *Proceedings of the 19th European Symposium on Artificial Neural Networks*.

LaValle, S. M. 1998. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Department, Iowa State University.

Lin, K.; Yang, H.-F.; Hsiao, J.-H.; and Chen, C.-S. 2015. Deep learning of binary hash codes for fast image retrieval. *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* 00:27–35.

Manning, C. D.; Raghavan, P.; and Schütze, H. 2008. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

Osborn, J.; Summerville, A.; and Meteas, M. 2017. Automatic mapping of nes games with mappy. In *FDG '17 Proceedings of the 12th International Conference on the Foundations of Digital Games*. ACM.

Sturtevant, N.; Bulitko, V.; and Buro, M. 2005. Automatic state abstraction for pathfinding in real-time video games. In *Abstraction, Reformulation and Approximation: 6th International Symposium, SARA 2005, Airth Castle, Scotland, UK, July 26-29, 2005.*, 362–364.

Turian, J.; Ratinov, L.; and Bengio, Y. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL '10 Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 384–394. Association for Computational Linguistics.

Yang, X.; Mei, T.; Zhang, Y.; Liu, J.; and Satoh, S. 2016. Web image search re-ranking with click-based similarity and typicality. *IEEE Transactions on Image Processing* 25(10):4617–4630.