N-Gram in Fighting Games

Kyle Pike, Ju-ve Chankasemporn

# 1 Introduction

Implementing Smart AIs into complicated scenarios such as Fighting Games can seem daunting, but there are easy and effective solutions.

For our research paper, we made a small fighting game based off a game Ju-ve created in the past. We then used N-Gram to implement an AI that learns from fighting the player and begins to effectively predict the player’s attacks and defenses. We wanted to see how easy and effective an N-Gram implementation would be at reading player attacks and choosing counterattacks.

# 2 What is N-Gram

~

There are multiple different N-Gram orders, with names based on how many pieces of data the N-Gram will look for to form a pattern. For example, a Bi-Gram is a type of N-Gram that looks at sequences of two pieces of data. A Tri-Gram will look at sequences of three, and so on. [1].

As the player takes new actions, the action history will delete, or “forget” the oldest action. This makes the N-Gram seem to have a sort of memory and will cause it to forget old strategies if the player hasn’t used them recently.

# 3 Game Overview

The simple fighting game we created, called “Walk Up Throw” is a simple fighting game with three main actions. “Attack”, “Block” and “Throw. Throws bypass blocks, blocks block attacks, and attacks outrange and out prioritize throws.

This makes the game feel like an advanced version of rock paper scissors, where there is not just strategy, but also skill.

# 4 AI Expectations

Our expectation for the AI was that it could predict the next action a player would take, then it would perform some sort of response to that action.

For example, if it anticipates the player will block, the AI would perform a grab to break through the block and defeat the player. If the player is anticipated to attack, the AI will block and then counterattack. Finally, if the player is attempting to grab, the AI will jab hit the player and bypass the grab.

The AI decides the predicted action by using a Bi-Gram to determine the likelihood a player will attempt a given action, then it uses RNG to determine a choice from those actions. This basic AI is a good testing ground for the effectiveness of the N-Gram implementation.

# 5 N-Gram in Our Game

We focused on implementing a Bi-Gram structure for our N-Gram project. In our game, the N-Gram looks at the first move of a sequence and tries to predict the second move in that sequence by referencing a 30-action long history of all of the players’ moves. It calculates the odds of the second move of the sequence appearing by checking how often it appeared in the move history. Finally, it uses RNG to randomly pick from the odds which move it thinks the player will use, and determines which strategy to use to counter it. The N-Gram structure allows it to learn from the player and adapt to overused strategies.

# 6 Hurdles to Implementation

We had to overcome a few hurdles while implementing the Bi-Gram AI into our game. One major hurdle was determining what counted as an “action”. Many fighting games require complex button inputs to perform moves, each input could be tracked separately, or we could simply track the outcome of the inputs. Both had benefits, tracking each input would allow our AI to predict multi-input attacks before players finished performing them. However, tracking each outcome would allow the AI to understand combos and interrupt them. Since our game was so simple, we chose to have each input be a separate button for ease of testing, and we tracked the outcome of the inputs. This choice allowed us to easily display the actions in the action log as well as the enemy’s prediction and the predicted chances of each player action. This implementation also allowed us to have a significantly smaller action log, which meant that it was easier to understand what data the AI was looking at.

Another choice we needed to settle on was how we wanted to implement how the AI would act when there was no action history to reference for a given move. Originally, the AI would just default to anticipating the player would attempt a throw. This didn’t seem like the best solution, so we altered it to assume that each action had an even chance if there was no history for that action. For example, in the sequence AGAB, our AI would predict an even split between Attack, Grab/Throw, and Block as the next player action.

# 7 Benefits of N-Gram

In our game, the benefits of N-Gram became immediately obvious. Our AI instantly became significantly more challenging for the majority of players that tested it, and it didn’t require preprogramming a significant number of odds and fighting data to achieve. All we needed to implement was a counter to whatever attack the AI estimated.

In more complicated fighting games, the ease of use of N-Gram would help programmers make challenging or interesting AI opponents for skilled players. Furthermore, N-Gram action logs could be locked-in to make an opponent with specific odds of reading given attacks. This means that developers could train their AI to anticipate specific combos more often than others by directly playing their games. They could even make weaker or stronger AI by changing the size and content of an action log, as well as the Order of the N-Gram that uses it.

# 8 Polish and Future Improvements

Our game could be tweaked and polished in a number of ways in the future.

-

# 1 Introduction

You must use this template doc as your starting point. Read the whole thing before writing, since it will save you time. Don’t change margins, spacing, fonts, or styles. Everything is set up to accurately measure page count.

This is the introduction. The introduction is the place to motivate your article and explain what will be covered. Typically this is done in one or two paragraphs, but occasionally authors will stretch it out to three or four. The introduction is incredibly important and should be refined or rewritten once the article is done to make sure it’s the best part of your article. From previous experience, the introduction and conclusion are often the weakest parts, which is unfortunate since this can cause readers to immediately stop reading your article.

The tone of your article should be expert and friendly. Carefully construct relevant examples that bring out the genius and relevance of the article. Try to inspire the reader to go off and actually use it. The article should be aimed at the intermediate/expert level, but still easy to understand. Strive for clear and concise sentences that are easy to digest. More people will benefit if they understand it the first time it is read. Above all, always keep a heavy slant toward practical game development (as opposed to academic research).

# 2 Headings and Styles

There are four levels of headings, but we recommend that you primarily stick with just the first two (readers start losing track of your hierarchy once they get to heading 3 and 4). Make sure you only use the “Styles” in Word’s ribbon bar (don’t change fonts or sizes). For example, this paragraph is written with the Normal style (Times New Roman 12 point). If you include references to code, like the class Route or the variable iCount, then make sure the particular code reference is in the Code style (Courier New 12 point).

When titling your headings, avoid overly cute or humorous titles. Yes, you are a funny writer, but we need to keep a level of professionalism. Subtle humor might be OK in limited amounts, but your section editor reserves the right to change or clean it up. It seems obvious, but make your titles helpful to understanding what follows.

# 4 Figures and Tables

## 4.1 Figures

(Note: The previous two consecutive headings without text between them is a problem. This is called a stacked heading and isn’t allowed. You’ll need to break these up with at least one sentence of normal text.)

Please embed figures directly into the Word file (but we MUST have the original figure files). Figures should be numbered starting at 1 and have a description that is a full and complete sentence (ending in a period). Refer to figures by name, as in the following example. Please refer to Figure 1.

Collision_Detection_and_Resolution_Figure02

Figure 1 Overlap testing is problematic for small, fast moving objects, like bullets.

Non-screenshot figures must be vector-based, such as using Word drawing capabilities (don’t use bitmaps unless they are screenshots). Any program that can produce vector-based art should work, as long as it can output a file in Adobe Illustrator (.ai), PDF, or EPS.

## 4.2 Tables

Tables differ from figures in that they have their description placed above the table. Tables should be numbered starting at 1 and the description should be a full sentence ending in a period. Please don’t make any fancy formatting within tables.

Table 1 Statistics gathered from past player moves.

|  |  |  |
| --- | --- | --- |
| Player Sequence | Occurrences | Frequency |
| Low Kick, Low Punch, Uppercut | 10 times | 50% |
| Low Kick, Low Punch, Low Punch | 7 times | 35% |
| Low Kick, Low Punch, Sideswipe | 3 times | 15% |

# 5 Code

If you are going to include code fragments, you’ll need to follow some simple guidelines:

1. Make sure that the code is using the Code style (Courier New 12 point).
2. Make sure there are NO TABS in the code (replace tabs with four spaces).
3. Make sure every line of code is 67 characters or less to avoid wrapping.
4. Class names, types, namespaces, attributes, methods, variable names, keywords, functions, modules, commands, properties, parameters, values, objects, events, XML/HTML tags, and similar elements inside the normal body of text should be listed in Courier New (12 Point) style.
5. Include the minimum amount of code required to get your point across. You can place additional code on the book’s website, so only the pertinent parts need to take up space in your article. The following is an example of some code.
6. Word will auto-capitalize the first letter of sentences, so watch out for this in your code.

## 5.1 Code Listings

You can have a couple lines of code right after a paragraph, but if you have too much, it should be a code listing, like Listing 1. The entire listing will be surrounded by a gray box in the final book.

Listing 1 Function to add one to an integer.

int AddOne(int a)

{

return a + 1;

}

# 6 Equations

If you want to highlight an equation, you can refer to it like Equation 1. Use the Equation Editor in the INSERT tab of the ribbon bar. You can place equations in-line in a paragraph, but it usually ends up making the paragraph spacing look weird, so avoid it if possible.

(1)

# 7 Editing

Your section editor and peers will help edit/comment on your article. When they do so, they will be using the “reviewing” tools in Word.

The amount of editing that section editors perform will vary based on your skill as a writer. If sections of your article are poor (poor English, confusing, unclear, etc.), those sections might be dropped or rewritten by yourself or the editors. If the entire article doesn’t meet the standards of the book, the decision to drop the article can be made by the section editor or series editor. This is rare, but it inevitably happens to one or two articles.

If your article requires heavy editing in places, section editors might help out, but you are free to adjust or rewrite these sections yourself. You’ll see the final version that will be printed so that you can approve or adjust any editing. In the past, most authors appreciate the help, if only to point out what was unclear so that they could rewrite it themselves. Others were grateful for having a fresh set of eyes understand the ideas that were trying to come out but needed a little help.

# 8 Conclusion

Your article should include a brief conclusion summarizing what the reader should have learned from the article. You can also motivate the reader to use the technique or give guidance for taking the ideas further.

# 9 References

A list of references is not absolutely mandatory, but it’s highly recommended. Below is a list of guidelines for each type of reference. When the reference in the text, be sure it is inside of a sentence and supports the ideas presented [Lewis 91]. Don’t refer to a bracketed reference directly as a noun.

## 9.1 Authored book:

[Woods 06] Woods, D.D. and E. Hollnagel. 2006*. Joint cognitive systems*. Boca Raton: Taylor & Francis.

## 9.2 Chapter in multiauthored book:

[Wiens 83] Wiens, J. A. 1983. Avian community ecology: An iconoclastic view. In *Perspectives in ornithology*, ed. A. H. Brush, and G. A. Clark, 355–403. Cambridge: Cambridge Univ. Press.

## 9.3 Journals:

[Terborgh 74] Terborgh, J. 1974. Preservation of natural diversity. *BioScience* 24:715-22.

## 9.4 Electronic journal:

[Testa 00] Testa, B., and L. B. Kier. 2000. Emergence and dissolvence in the self-organisation of complex systems. *Entropy* 2, no. 1 (March): 1-25. http://www.mdpi.org/entropy/papers/e2010001.pdf.

## 9.5 Unpublished Documents:

[Schwartz 00] Schwartz, G. J. 2000. Multiwavelength analyses of classical carbon-oxygen novae. PhD diss., Arizona State Univ.

[O’Guinn 87] O’Guinn, T. C. 1987. Touching greatness. Paper presented at the annual meeting of the American Psychological Association, New York.

## 9.6 Online Documents:

[Adamic 99] Adamic, L.A., and B.A. Huberman. 1999. The nature of markets in the World Wide Web. Working paper, Xerox Palo Alto Research Center. http://www.parc.xerox.com/istl/groups/

iea/www/webmarkets.html (accessed March 12, 2001).

[U.S. Census Bureau 00] U.S. Census Bureau. 2000. Health insurance coverage status and type of coverage by sex, race, and Hispanic origin. Health Insurance Historical Table 1. http://www.census.gov/hhes/hlthins/historic/hihisttl.html.