Department of Computer Science
The University of Auckland
New Zealand

Using Case Based Reasoning to Price Items in a Virtual Economy

Andrew Ettles

October 2016

Supervisor: Ian Watson

A thesis submitted in partial fulfillment of the requirements of Bachelor or Science Honours in Computer Science

Abstract

Virtual economies in massively multiplier online (MMO) video games provide a self-contained environment with few exogenous influences in which we can observe microeconomic behavior of individual agents. In this dissertation we observe the decisions players make when selling in-game items in the action role playing game (ARPG) *Path of Exile*. Each player in the game is their own vendor of items within the global in-game marketplace. To engage in trade players need to be able to identify and price items worth selling. We show that optimally pricing items, especially if items are heterogeneous, can be a difficult task for players. With time and knowledge constraints players resort to using heuristic pricing strategies.

To help solve this "pricing problem" we design and implement a novel artificial intelligence (AI) system using case based reasoning (CBR), a methodology that has commonly been used for cost estimation or valuation. Given an unpriced item, the system uses historical market data of similar sold items to formulate a fair market price. Depending on the characteristic properties of the item different CBR models are considered. An internal evaluation of the system is conducted using cross-validation. The results of the evaluation are then used to optimize the parameters of the CBR system.

Contents

1 Introduction

1.1 Introduction

With the rise of Massively Multiplayer Online video games (MMOs) has come the advent of the virtual economy. A marketplace in the virtual game world where human players trade virtual goods and currencies. Virtual economies are increasingly interesting to economists as the interactions between agents within these worlds is in many ways the same as in "real-world" economies [2]. Virtual economies lack the complexity and array of exogenous influences that can be hard to model in the real world. Because of this many virtual economies have been used as a *sandbox* to build and test economics models.

One question worth investigating in any economy is how individual agents set the price of goods they are selling? This dissertation investigates this microeconomic decision problem in the MMO action role playing game (ARPG) *Path of Exile*, developed by the New Zealand based company *Grinding Gear Games* (GGG). In the *Path of Exile* economy each player is their own vendor. Due to the large diversity in items within the game, most found items will have little value to each player but could be valuable to another player. To quote the old adage *"One man's trash in another man's treasure"*. Selling items allows players to get currency which they can use to purchase items that are beneficial to their own virtual character.

While playing the game, players will find numerous different items. Each player has to be make a decision on whether an item is worth keeping, selling or discarding. If they believe the item is worth selling to another player then they can list the item on the market for a set price. Due to the heterogeneous nature of many items determining a fair market price can be a difficult problem for players. Players can use past trading experiences or comparisons to similar items on the market to formulate a valuation. If items are difficult to price then many items will be frequently listed above or below market price, resulting in an inefficient market. We investigate the methods players commonly use to price different types of items and the difficulties that are entailed.

To help solve the pricing problem that players face we design and build a novel case based reasoning (CBR) system for predicting the fair market price of an item. The CBR methodology is a popular artificial intelligence (AI) process of solving problems by analyzing and adapting solutions to previous *similar* problems [12]. CBR simulates aspects of human memory and inference whereby we base our current decisions to problems on previous experiences with similar problems [20]. CBR is generally more robust, transparent and a faster learner than other machine learning techniques [21]. Several cost estimation and valuation systems have been implemented using CBR [14, 13, 15] and [23, 24].

Our CBR system takes unpriced items and searches historical market data for similar sold items. Using a combination of the prices of these similar items we can get an estimation for the price of the unpriced item. We describe in detail the process and challenges of designing a CBR pricing system for game items, as well as ways of evaluating and optimizing the system. Through analysis of market data and designing the CBR system we gain insight into the microeconomic nature of the different types of items that are sold in *Path of Exile*. Findings from this dissertation may then generalize to other virtual economies and perhaps even self-contained economies in the real-world with similar characteristics.

1.2 Related Work

1.2.1 Virtual Economies

The majority of the literature on virtual economies focuses on the real-world economic ramifications that virtual economies can provoke. Real-money trading (RMT) and gold farming involve exchanging virtual currencies and items for real world currency [25]. Players that engage in RMT usually do so to gain an economic advantage over other players within the game environment. RMT is against the terms of service in most popular MMOs (such as World of Warcraft) but can be hard to detect so it remains a prominent issue. Some game developers do not see this as an issue and have facilitated RMT within their games as an alternative way to generate revenue. The most notable example is that of of *Diablo III* which controversially implemented a real money auction house (RMAH) to purchase in game items for real-world currency [26]. The RMAH auction house was very unpopular within the player community. As more and more players left the game the RMAH was scrapped in September 2013 (which no inter-player trading as the replacement) [27]. In *Path of Exile* RMT is against the terms of service and no items are purchasable with real money that could translate into an economic advantage for a player.

Salter and Stein [28] examined monetary emergence in the game *Diablo II*, which is commonly referenced as the game *Path of Exile* is based upon. In *Diablo II* the in-game currency gold did not have the characteristics to serve as a monetary unit for inter-player exchange. As a result Salter and Stien [28] document the emergence over a period of time of alternative standardized currencies within a barter economy. This emergence process also occurred (albiet over a shorter timespan) in *Path of Exile* where “Exalted Orbs” and “Chaos Orbs” are accepted by the community as the standard currencies to buy and sell items in game.

1.2.2 Methods for valuation or cost estimation

Mathematically we can think of pricing an item as a function of a set of explanatory variables X.

$$price = f(X)$$

X consists of the properties of the item that influence the price, as well as the time it is being sold. There are a number of possible methodologies that can be applied to solve this problem, each with it’s own set of strengths and weaknesses.

Regression Models

Econometrics is a broad branch of economics that relates empirical data to economic models. The fundamental tool for creating these models is regression analysis, a means of estimating a dependent variable (price) based upon a set of predictors (or explanatory variables) X. Multiple regression analysis can be represented by:

$$price = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n$$

where $\beta_0, ..., \beta_n$ are the set of coefficient estimators that the regression analysis tries to optimize. Regression techniques have been popular tools for valuation and cost estimation since the 1970’s [13, 30].

Statistical Regression models are not good for dynamically changing data [13]. Inflation/deflation occurs over time for items. A regression model can not effectively adjust for fluctuations in the price of items. Regression models are also very difficult to accurately build when there is large number of variables. Especially if there are interdependencies between these variables, heteroskedasticity and non-linear relationships [13]. Rossini showed in the context of residential property valuation that these statistical methods are outperformed by artificial intelligence approaches [29].

Rule based systems

Expert systems that use rule based reasoning leverage domain knowledge of human experts. This specialist knowledge is transfered to rules within the system which emulate the decision making abilities of the human expert. Several rule based systems have been implemented for property valuation [31].

A simple query based expert system called *Durian* [5] has been implemented for finding specific items on the *Path of Exile* market. *Durian* takes as input simple queries such as "2H axe 450pdps 1ex". *Durian* then continuously searches the market for two handed axes with more than 450 physical damage per second (DPS) and are priced at 1 Exalted Orb or less. If an item that matches the query is found then the player is alerted. The problem with this and other expert systems is that extracting expert knowledge (writing realistic queries) is a time consuming process. Also item prices fluctuate over time so rules must be updated regularly to maintain the accuracy of the system.

ANNs

Artifical Neural Networks (ANNs) simulate the neuron activity within the human nervous system. Since the 1990s, ANNs have been commonly used for cost estimation [13, 33] and valuation [24, 32]. ANNs have an advantage over other AI approaches in that they require little modeling and parameter setting from the architects of the system while still providing high predictive accuracy. However neural networks require a lot of data to train the system, with learning being a time-consuming process [13]. Neural networks have also been described as a "black box" because they provide little explanation to how they reached their solutions [24].

CBR systems

A detailed overview of how a CBR system is built and operates is provided in Chapter 3.

As specified in the introduction CBR systems have also been popular tools for cost estimation and valuation. As opposed to ANNs, CBR is more intuitive and allows for better justifications of solutions to problems. By referencing previous similar cases, users of the CBR system can easily see how the solution to the current problem is reached. CBR also tends to be more robust to changes in data of time compared to the above methods meaning that it is more practical for long-term use [13]. Once a sold item has been seen it can be added to the case base, no time-consuming retraining of the system has to be done. To ensure the case base does not become stale old cases can be discarded over time. A problem with CBR systems is that they are reliant on having sufficient coverage over all feasible cases. If no similar cases exist in the cases base then it is unlikely we can infer an accurate solution. This is also a problem if the quality of case solutions is poor [11].

2 Path of Exile

2.1 Path of Exile Economy

Path of Exile is a an online ARPG heavily inspired by the *Diablo* series, notably *Diablo II*. The game has been designed with a large emphasis on character itemization. Items (or loot) are obtained by killing virtual monsters within the game. Equipment items, which can be worn by player characters, contain mods (short for modifiers) which make the character more powerful. Other items such as currency items can be used to randomly alter the properties of other items. Almost all items can be traded with other human players. Players will find many items while playing, most of which will not be useful to their own character but may be beneficial to other players. Trading allows players to exchange these unwanted items for currency items, which can then be used to purchase items that are beneficial to the player's own character.

2.1.1 Currency Items

The game does not have a pre-determined currency (like gold in the Diablo games), instead a multitude of currency items exist. Currency items are commonly used to craft other items, for example an Exalted Orb adds one extra mod to a rare item (up to a maximum of 6 mods). Currency emergence [28] was a relatively quick process in *Path of Exile* as the developers designed these currency items, which had intrinsic value within the game, to be also used as standard currency for trading between players.

The most commonly used currency items for trading are Chaos Orbs and Exalted Orbs, which have been accepted by the community as standard. Over 90% of items on the market are priced using either of these two currencies. As a result these two currencies are the most liquid. Currencies have exchange ratios which fluctuate over time. For example the Chaos Orb:Exalted Orb ratio usually sits at around 70:1.

For the purposes of this project all item listings are converted and priced in Chaos Orbs using a set of fixed exchange rates. This assumes that the exchange rate between Chaos Orbs and other currency items is relatively stable over time. Usually after the first few weeks in a new league (a period of around 3 months with a fresh economy) the exchange rates stabilize [6].

2.1.2 Trading System

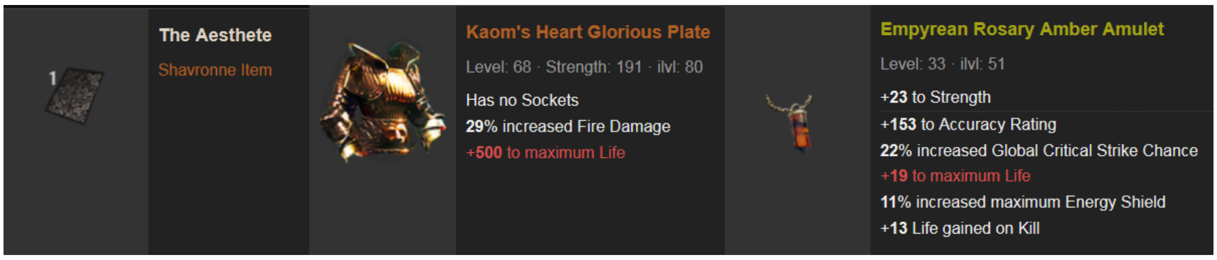
There is no in-game system for viewing items that have been listed on the market, instead one must use a third-party tool. The most popular website for searching for items is the website *poe.trade*. To emphasize how commonplace *poe.trade* is for trading, it is among the top 10,000 most visited websites in the world (top 4,000 in the United States) according to Alexa rankings [34]. *poe.trade* indexes all items that have been made *public* in a player's in-game stash (a finite amount of storage space for items you don't need to carry on your character) through the *Path of Exile* public stash tab API.

When purchasing items players can use filters within *poe.trade* to search for specific items. Found items can be displayed with a fixed purchase price or no price set at all (although the vast majority are set at a fixed price). To purchase the item you must message the player in game. They must then invite you to their party where you must convene at a common location in-game to trade the items. To sell an item, you must place it in a public tab within your item stash. The *poe.trade* indexer will automatically (within seconds) display the item on the market. A buyout price is set by right clicking the item. A seller is not held to the price that they have listed an item on the market.

The argument GGG has against implementing an in-game auction house in Path of Exile is that they want trading to incur a *temporal cost*. That is it takes time and effort to buy and sell items, the opportunity cost of trading is playing the game to find more currency and items. This helps prevent an over-supply of poor quality items being placed on the market. It also prevents automation of trading, whereby bots can continually scan the marketplace and instantly purchase any items that have been listed below market price. GGG did not want to repeat the mistakes of Diablo 3 where an over-supply of items on the auction house led to massive deflation in the value of gold, the standard currency used by players to trade [27].

2.2 Types of Items

We class the commonly traded items in *Path of Exile* into three categories: Commodites, Uniques and Rares. Figure 2.1 gives an example of each of these three items.


1
The Aesthete
Shavronne Item

Kaom's Heart Glorious Plate
Level: 68 · Strength: 191 · ilvl: 80
Has no Sockets
29% increased Fire Damage
+500 to maximum Life

Empyrean Rosary Amber Amulet
Level: 33 · ilvl: 51
+23 to Strength
+153 to Accuracy Rating
22% increased Global Critical Strike Chance
+19 to maximum Life
11% increased maximum Energy Shield
+13 Life gained on Kill

Figure 2.1: Examples of the three categories of items. From left to right: commodity, unique, rare

Commodities

Commodities are items which are homogeneous. Examples of commodities are currency items and divination cards. Commodity items should never be discriminated against. When purchasing a commodity a rational player should always buy the cheapest available commodity on the market. Because of this there is normally little variance in the price of commodities. Scarce commodities may be vulnerable to price fixing whereby a player purchases all of the items on the market and the sets the price well above the market value.

Unique Items

Unique items are typically powerful pieces of equipment that tend to define certain character builds. Unique items have a set of fixed item mods. Some of these mods may have a variable range of values. In this sense, unique items with the same name may differ in price. We say a unique is well-rolled if the value of the variable mods is close to the maximum in it's range. For example, the Kaom's Heart unique item has 600 armour, +500 life and Increased Fire Damage ranging between 20-40%. A Kaom's Heart with a 40% Fire Damage roll will be worth more than one with a 20% Fire Damage roll.

Rare Items

Rare items consist of a selection of 4-6 mods from a pool of affixes. Each rare item can have up to 3 prefix mods and 3 suffix mods. These mods tend to lie within a relatively large range of values. It is extremely rare to find two rare items that are exactly the same. Because of this we class these items as heterogeneous. As rare items are almost always different it can be much more difficult for players to assess the value of the item.

2.2.1 How do players price items?

When a player comes across an item they believe to be worth selling they can go about listing it on the market. Setting the price of the item can be a difficult task.

Many inexperienced players do not that have the knowledge built up from experience trading a range of items. These players often will not be able to tell whether a item is valuable or not by sight. Without spending time researching the price by looking at similar items, they may list the item for far less than it is worth or even just discard the item. Because of the wide range of character builds and item combinations few players have a full knowledge across the entire market. Those that do have this knowledge can exploit this by identifying (usually via scripts) items that are priced well below market value. Traders that engage almost exclusively in this behavior are known as “flippers”. Flippers are common in in many virtual economies where markets are often inefficient and market data is easily accessible. Experienced players commonly claim the best way to make currency in the game is via flipping items [8].

Trading is costly. It takes time to price and list an item, then to complete a transaction with a buyer both players are required to meet at some common point within the game world. This transaction process takes a minimum of 20 seconds but is usually longer. Most players realize that the opportunity cost of trading is earning currency and finding new items by playing the game. Therefore most players will not sell items for less than some minimum price threshold. This minimum price threshold is dependent on the individual. Generally we would assume that wealthier or higher leveled players will have higher minimum price thresholds.

Most players employ some sort of **pricing strategy**. In economics a pricing strategy usually refers to a firm’s method of setting a product’s price to maximize some market objective such as profit, sales or market share. In this context, player’s pricing strategies reflect personal game objectives. For example, some players are aversive to trading and will only sell items that are valuable or even play “Self-found” whereby they will never engage in trade with any other player. Some common pricing strategies are:

1. **Undercut** - Find the lowest priced similar item on the market and set price slightly lower. Need to be aware that you are not undercutting an item that is under-priced relative to other items.

2. **Price Decay** - Set the price higher than what you think the item is worth based upon similar items on the market. Periodically lower the price of the item by a set amount until it sells.

3. **Approximate** - Price an item without looking at similar items on the market. Base price on heuristics and similar past transactions. If the item sells instantly then you

probably set the price too low, so next time you find a similar item put it up for a higher price. If an item isn't selling then you probably overpriced the item.

In practice players will likely employ a combination of the above strategies (or others) when selling items.

Undercut works well for homogeneous goods such as currency or unique items as buyers will almost always opt for the lowest price on the market. Undercut gives high turnover but results in a lower potential revenue as items are sold below market price.

Price decay works well for goods where the price of the item is uncertain. This is often the case for rare items where the large variability between items means finding similar items can be difficult. Turnover tends to be low as items are deliberately set at prices higher than predicted. The slower the rate of price decay the more likely the item will sell at or above market value.

Approximate works best for players with a wide market knowledge. Their trading experience creates a "case base" of past transactions which they can use to infer the price of items. Approximate results in the lowest amount of time spent pricing the item but can result in lower potential revenue (if items are potentially under-priced) or lower turnover (if items are overpriced). This method is vulnerable to fluctuations in item prices.

2.3 Data

Datasets of market data were collected from the now defunct Exile-Tools API [9]. The API indexed all items listed for sale by players using the *Path of Exile* stash tab API service. The indexer also stores historical data of items that are no longer available on the market. Data could be queried using Elasticsearch query DSL and returned in JSON format. Appendix 3 shows the JSON representation for a unique Windripper item.

2.3.1 Key item properties

- **id** - Unique id given to every item listed on the market.
- **item name** - Name of the item.
- **mods** - List of all the numerical stat values on an item.
- **price** - Amount of currency the seller wants for the item, can be unspecified.
- **added** - Time-stamp when the item was added to the market.
- **updated** - Time-stamp when the price or status was last changed by the seller.

- **status** - Whether the item is no longer on the market ("GONE"), or still on the market ("YES").
- **seller** - The account of the player selling the item.
- **attributes** - Other item properties. Some examples:
  - **item type** - Is the item a bow, ring, jewel, currency item...
  - **identified** - Unidentified items do not have their mods visible.
  - **rarity** - Normal, magic, rare or unique. Determine the frequency at which the items drop within the game.
  - **base item name** - Within equipment types different base items exist - e.g. Harbinger Bow, Imperial Bow, Maraketh Bow
  - **league** - Each league has it's own economy. Items can only be traded within the specified league.

2.3.2 Data Description

Data for five different items were sourced from the Prophecy Challenge league between 25/6/2016 and 20/7/2016 for the purposes of analysis. This included one item that is classed a commodity, one item that is classed as unique and three items that are classed as rare.

Items without a specified price are excluded as they give no knowledge of what they sold for. In the datasets less than 5% of items did not have a specified price. Items that are unidentified are also excluded from analysis except in the commodity case. Unidentified items could be classed as a gamble where the expected outcome is the market price of the average item. In the commodity case every item is the same so whether the item is identified or not is irrelevant. As mentioned previously all prices are converted into a Chaos Orb equivalent price using fixed exchange rates.

We assume that if an item has status "GONE", it has sold for it's listed price at a time given by the updated time-stamp. The glaring issue with this data is that sellers are not held to their prices. A seller may accept an offer from another player for an item for lower than the listed price. Sellers may also remove the item from the market altogether, in which case the status of the item is "GONE" but we cannot tell whether or not the item was sold or not. If a listed item remains on the market at the player's minimum price threshold for a long period of time they may conclude the item will not sell and therefore get rid of the item to free up stash tab space for selling other items. We assume that if an item has status "GONE", it has sold for it's listed price at a time given by the updated time-stamp.

2.3.3 Data Pre-processing

The JSON datasets where parsed into condensed single table, comma separated value (csv) files containing only the attributes that were determined to be relevant for analysis. All datasets contained item id, item seller, updated time and price.

Duplicates

Every time an item is listed under a new seller it generates a new unique id. This means that the same item could appear multiple times in the dataset. For instance if player A sold an item to player B then at a later date player B sold the same item to player C for a different amount. Duplicates are not removed as they represent different transactions occurring at different time periods. The same item being sold multiple times improves the accuracy of the pricing system by getting a truer estimate of the market price of the item (assuming transactions are independent from one another).

Outliers

Outliers should be identified and removed. Outliers in this context do not refer to items that do not share many attributes with any other items but instead refer to items that have obviously not been sold at the correct price. This can include well under-priced items that are bought up quickly by flippers. Usually the time difference between being added to the market and removed from the market is small. Also items that are well-overpriced and then removed all together from the market will show as being sold.

Outliers can severely hinder the accuracy of a case based system [11]. Take for instance a simple system that returns as a solution the price of the most similar item. If the most similar item is grossly mis-priced then that error directly propagates to the case solution. Obviously we consider multiple similar items when recommending a price for an unknown item. However, taking a weighted distribution (such as the mean) over the items means that an outlier can still heavily skew a solution. This can be partially solved using methods such as a median or trimmed mean (described later).

By eyeballing the dataset obvious outliers can be removed. This is relatively easy for commodities and unique items where it is apparent what the range of acceptable prices is. System evaluation using cross validation can be used to further detect and remove outliers (described later).

3 CBR System

3.1 CBR Model

3.1.1 CBR basics

Case base reasoning consists of solving new problems by adapting solutions to similar old problems. In the context of items in *Path of Exile* the problem is finding a fair market price for an unpriced item. The previous cases consist of historical market data of sold items, with the price they sold for being the case solution. The basic CBR cycle [11] consists of:

1. **Retrieval** - Given a target item, find a set of similar sold items from the dataset. Similarity between items is based upon the item's properties as well as how long ago the item sold.

2. **Reuse** - Adapt the prices of the returned similar items to formulate a price for the target item.

3. **Revise** - Evaluate the quality of the solution. This could involve listing the target item on the market at the predicted price although there are other methods we can use for evaluation. See the evaluation section for more details on these methods.

4. **Retain** - If the item sold at the predicted price then it will be automatically added to the case base (historical market data) for future problem solving.

3.1.2 Retrieval

To retrieve the most similar items we use the k-Nearest Neighbours (KNN) algorithm. KNN compares the target item to every item in the case base calculating the similarity between each item. KNN then outputs the k items that have maximum similarity to the target item.

3.1.3 Similarity

Similarity between items or global similarity can be represented by the following function for a target item T and sold item S [12]:

$$Similarity(T, S) = \sum_{i=1}^{n} f(T_i, S_i) \times w_i$$

where n is the number of attributes deemed to be influential to the price of an item, i is an attribute and w_i is the weight (or importance) of that attribute. f represents a local similarity function, a way of determining how similar two attribute values are. The global similarity can be seen as a weighted sum of the local similarity functions for all attributes. The design of individual local similarity functions are specific to each item and are described later.

3.1.4 Adaptation

Having a list of the k nearest neighbours we have can use the prices of these items to formulate a prediction for the price of the target item. Methods we could use include taking the mean price, taking the median price or taking a weighted average (such as a Gaussian distribution) over the prices based upon the order of similarities.

The problem with taking a mean over the set of prices is that the quality of solutions is not guaranteed, so outliers can heavily skew the result. A Gaussian distribution has the same problem if the most similar item returned has a poor solution. The median price is more robust against outliers but neglects the distribution of the set of prices.

Because of this we use a trimmed mean method where we exclude the highest and lowest 20% of prices. Note if the value of k is either 3 or 4 then we exclude the highest and lowest prices. Different adaptation methods may suit different datasets better than others. Techniques such as cross-validation could be used to evaluate the performance of such methods.

3.1.5 Attribute Weights

Determining the correct weights for a set of attributes is often considered to be a pseudoscience but is very important within a CBR model [17]. The distribution of weights directly affects the similarity between items and hence the target solution returned. For example, if an item has a weight distribution with all weights being zero and the time similarity having weight 1, then the case base will only return the k most recently sold items (as with a commodity).

Subjective Methods

Normally, relative weighting of attributes is determined by experts within the domain who can intuitively tell the order of importance of attributes [11]. These methods for determination are known as subjective methods [16]. Subjective methods suffer from a lack of adaptability to changes in the underlying data. Also, determining how much more important an attribute is compared to another has been shown to be difficult by human estimation [14, 13]. For example, in *Path of Exile* every few months new content in released along with new skills/items and game balance changes to existing skills/items (commonly referred to as to as buffs and nerfs). This may cause some specific item attributes to increase or decrease in value. Hypothetically if all fire spells were given a blanket 50% increase to damage, then players will be more inclined to play with these skills as they are more powerful. This means that items with an “Increased Fire Spell Damage” attribute will increase in demand, so the attribute importance (weight) should be higher (ceteris paribus). If relative attribute importance is changing often then the weight model will have to be updated frequently. Subjective methods fail when experts are unable to observe and correct these changes in a timely manner.

Objective Methods

Conversely, pure computational approaches or objective methods can be taken to attribute weight optimization. Several such approaches have been proposed in recent years including genetic algorithms, simulated annealing, linear programming and artificial neural networks [16]. All these techniques being common algorithms for optimization and learning within the artificial intelligence discipline. Objective methods have the advantage of not requiring any subjective input from human experts. However, these approaches have limited capability when data is substantially noisy - in which case reverting to expert knowledge can prove to be more reliable.

Hybrid Approaches

The above objective methods tend to search for globally optimal weight distributions. However if we have reasonable initial values set by subjective methods, then hill-climbing or gradient descent methods can quickly converge to local optima, which represent a "good enough" weight distribution [16]. This may or may not be the global optimum weight distribution.

The method we use for deciding weights for each item attribute is based upon a hybrid approach. We use "expert" knowledge backed up by statistical methods to decide reasonable approximate values for initial weights. After this we using a basic hill climbing method. For each item attribute we incrementally alter the solution until we come to a "hill", a value whether the objective function is maximized relative to the values either side of it. In our case the objective function is minimizing the average error in leave one out validation across the entire dataset (see Evaluation section for details). This method differs slightly for different item types depending on the size and dimensionality of the dataset, the noisiness of the dataset as well as whether the weight distribution of the data is likely to change over time.

3.2 Implementation

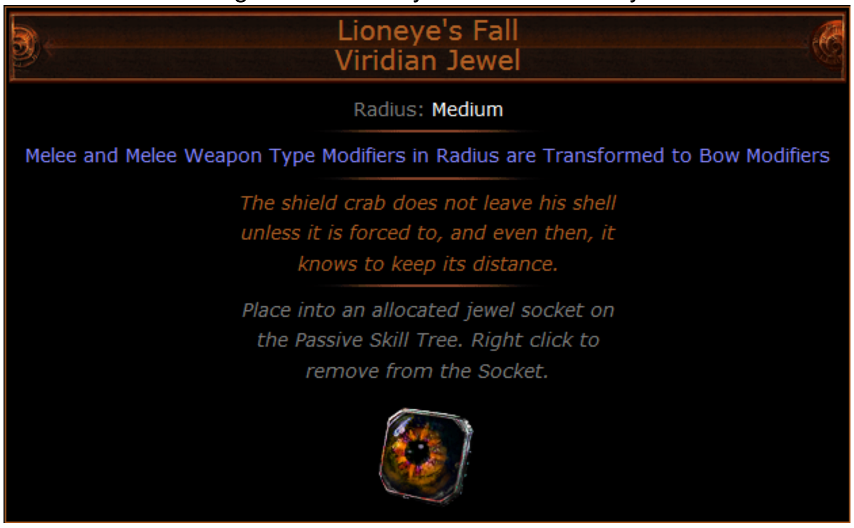
Initially we tried using open-source tools *myCBR* and *jCOLIBRI*, two popular frameworks for building CBR applications. However, these tools were inflexible for the necessary adaptations to the CBR model that were required for rare items. We settled on building the CBR system from scratch as a C# console application.

This implementation builds the case bases from the given datasets at runtime to be stored in memory. In future implementations each case base should be stored in a separate database.

3.3 Commodity (Lioneye's Fall Viridian Jewel)

On face value pricing a commodity using a CBR system may be uninteresting and even suboptimal. The most efficient way to price a commodity is to undercut the lowest available on the market. This means that a buyer of that commodity should always purchase your item over others. This assumes that the market for commodities is somewhat efficient and under-priced commodities do not remain on the market. The only variable attribute for a commodity is the time it was sold. Therefore, all the CBR system tells us is

Figure 3.1: Lioneye's Fall commodity



Lioneye's Fall
Viridian Jewel

Radius: Medium

Melee and Melee Weapon Type Modifiers in Radius are Transformed to Bow Modifiers

The shield crab does not leave his shell unless it is forced to, and even then, it knows to keep its distance.

Place into an allocated jewel socket on the Passive Skill Tree. Right click to remove from the Socket.

what price the commodity sold for on average at different periods in time. Assuming that un-priced items always come in at the present time, the CBR system returns a weighted combination of the k most recently sold items.

3.3.1 Dataset Analysis

The Lioneye's dataset has $n = 607$ items. 6.59% of items are still on the market. Large

Table 3.1: **Sample statistics for Lioneye's Fall**

	mean	**SD**	**median**
Sold	14.87	7.31	13
On Market	32.01	29.54	27

standard deviations show how noisy prices can be, even for a commodity item. The mean price of Lioneye's Fall on the market is more than double those that have been sold. If we only consider those that have been sold within the past 12 hours ($n = 26$) the mean price is 10.57. A possible explanation is that most of Lioneye's fall items on the market are over-priced. The majority never get bought as their price is never the minimum (especially since the price trends down over time). At any point in time few to no under-priced items are present while several over-priced items are present. The mean price of items on the market has little connection to the market price. A better indication would be price of items that have sold recently.

Figure 3.2 shows the price of all Lioneye's fall in the dataset which sold. Figure 3.3 shows the same data after removing outliers (17 removed). This particular item shows a clear downward trend in price over time, which stabilizes around 5/7/16. Although we don't have data between 11/7/16 and 19/7/16 we can assume the trend continues as

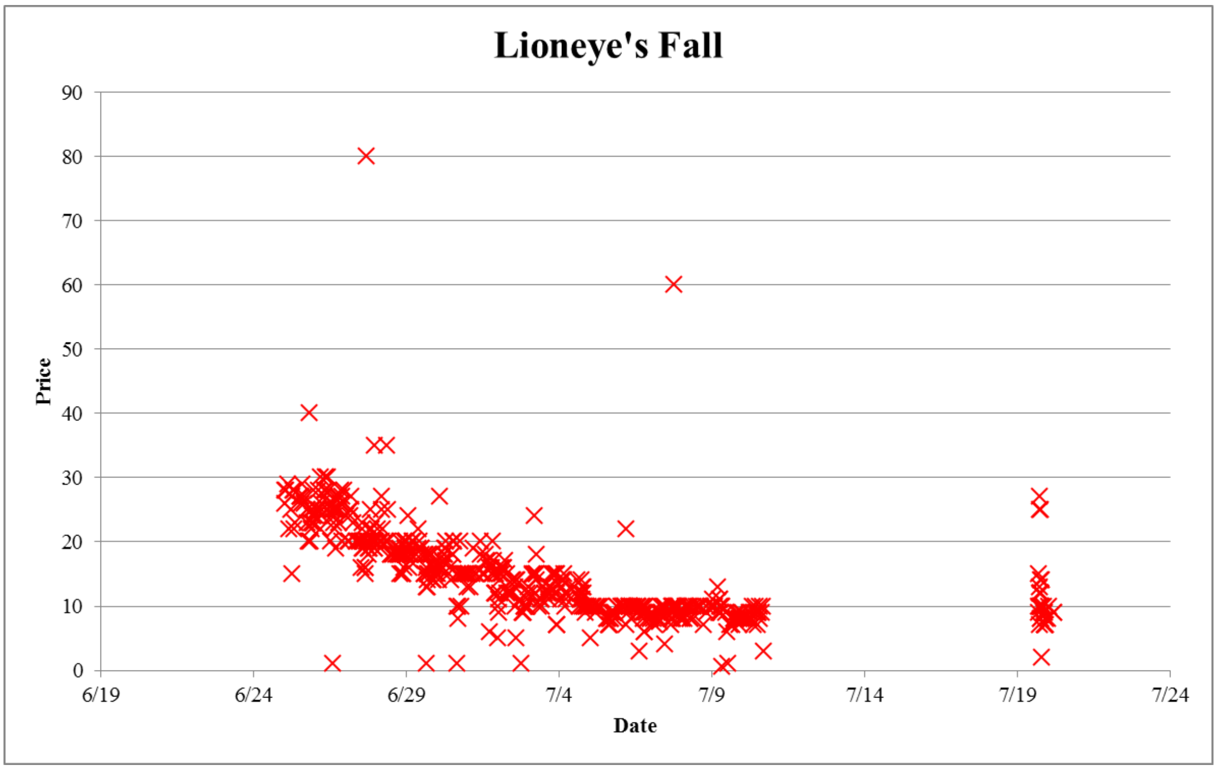


Figure 3.2: Sold Lioneye's Falls over a one month period

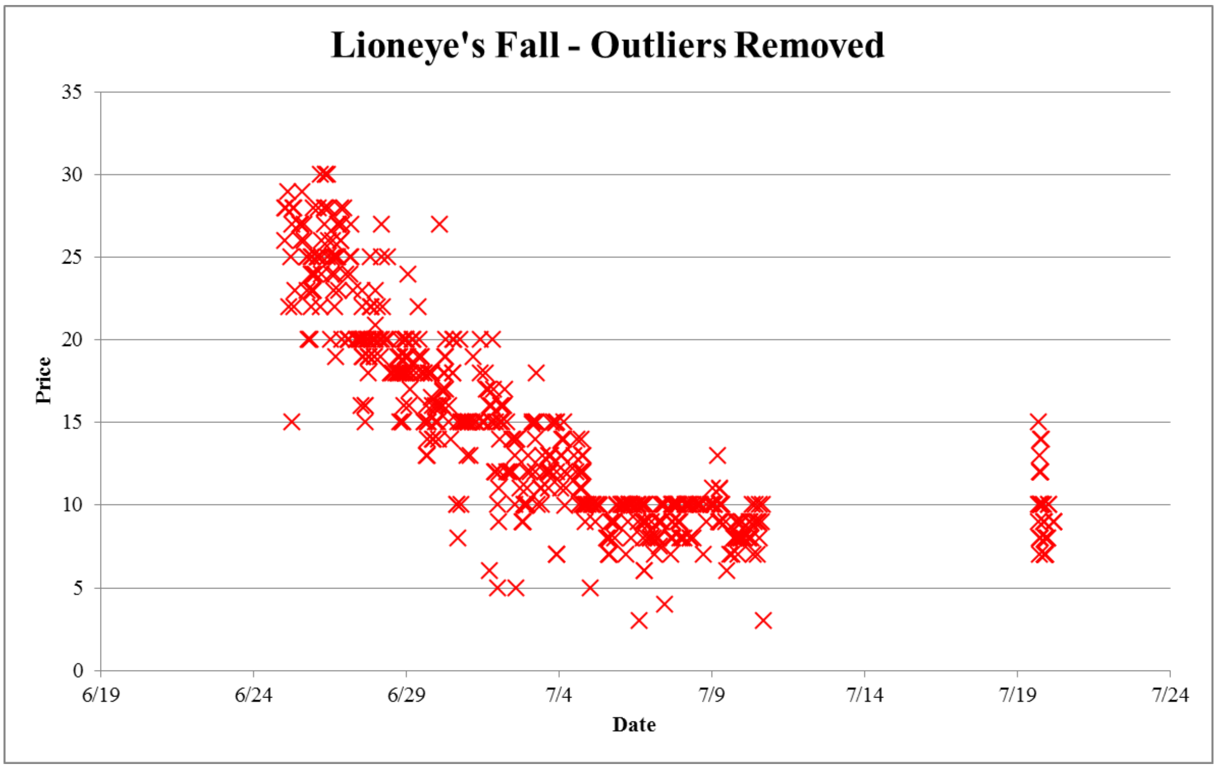
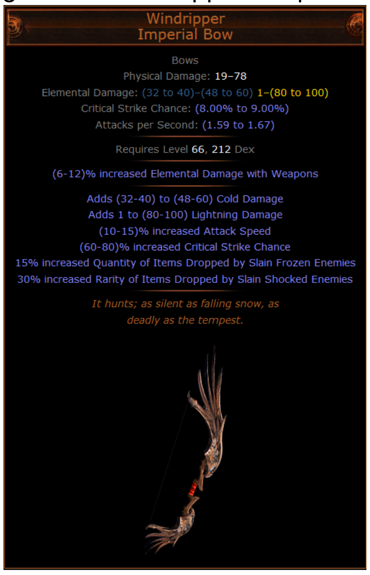


Figure 3.3: Sold Lioneye's Falls over a one month period with outliers removed

the average price either side of the interval is approximately the same. An interesting observation is that the most frequent prices tend to be multiples of 5 (10, 9, 20, 15 in order of frequency). This observation also occurred in the other datasets for items with prices greater than or equal to 10. This is consistent with previous research in pricing psychology where humans have a natural affinity towards rounded numbers (as our number system is in base 10) [10].

3.4 Unique Item (Windripper Imperial Bow)

Figure 3.4: Windripper unique item



Windripper
Imperial Bow

Bows
Physical Damage: 19–78
Elemental Damage: (32 to 40)–(48 to 60) 1–(80 to 100)
Critical Strike Chance: (8.00% to 9.00%)
Attacks per Second: (1.59 to 1.67)

Requires Level 66, 212 Dex

(6-12)% increased Elemental Damage with Weapons

Adds (32-40) to (48-60) Cold Damage
Adds 1 to (80-100) Lightning Damage
(10-15)% increased Attack Speed
(60-80)% increased Critical Strike Chance
15% increased Quantity of Items Dropped by Slain Frozen Enemies
30% increased Rarity of Items Dropped by Slain Shocked Enemies

It hunts; as silent as falling snow, as deadly as the tempest.

The unique item Windripper has four variable modifiers (excluding the implicit modifier which is deemed to have little effect on the price of the item). When found each mod varies uniformly within the it's listed range (shown in Figure 3.4). Players often search for weapons based upon aggregations of similar mods. We can combine Cold Damage, Lightning Damage and Attack Speed to calculate the elemental damage per second (Edps) of the item.

$$Edps = (Cold + Lightning) \times AtkSpd$$

Aggregation is a simplification of the model which reduces dimensionality. To test the sensibility of this assumption we could compare prices of Windrippers with similar Edps but contrasting values for cold and lightning damage. For example, a Windripper with 50 cold damage and 40 lightning damage vs a Windripper with 40 cold damage and 50 lightning damage, both with 10% attack speed giving the same Edps. We would expect that on average the prices of such items would be the same.

Weapon items in Path of Exile contain sockets and links which determine the number of skills gems which can be supported by the item. A bow item like Windripper can have a maximum of 6 sockets/links. A 6 linked item is highly sought after but hard to obtain. 5 linked are also sought after and hard to obtain but to a lesser extent. Items with4 links or less are trivial to obtain. In terms of price this means that the similarity between 6 linked, 5 linked and ≤ 4 linked Windrippers is almost zero.

3.4.1 Dataset Analysis

The Windripper dataset has $n = 748$. Order statistics are grouped by number of item links as there is almost no similarity between 6 link, 5 link and ≤ 4 link items. Similar to the commodity case, items on the market display higher prices than those that have been sold. Standard deviations are also large but this is more understandable considering unique items are heterogeneouss.

Table 3.2: **Sample statistics for Windrippers**

Windripper	n	**mean**	**SD**	**median**
6 Link Sold	54	513.61	101.91	480
6 Link On Market	5	627.2	88.45	640
5 Link Sold	192	45.44	28.82	39.5
5 Link On Market	21	60.48	24.60	50
≤ 4 Link Sold	440	18.49	18.15	15
≤ 4 Link On Market	36	37.34	24.90	28.5

3.4.2 Local Similarity

Five attributes are deemed influential to the price of a Windripper: critical strike chance, elemental dps, number of sockets, number of links and time sold. Local similarity functions need to be modeled for each attribute.

Range Similarity - Range similarities are defined as attributes values that vary uniformly between a set range. For example critical strike chance varies between 60-80 on Windrippers. We can design a particular local similarity function by considering the following questions [22]:

1. **Is it symmetrical?** For example, is a critical strike chance of 70 just as similar to 75 as it is to 65?

2. **Is it proportional or absolute?** Is the similarity between two items with attribute values 60 and 62 the same as two items with attribute values 78 and 80 (absolute similarity). Or is the similarity between two items proportional (e.g. 60/66 and 70/77, both have a 10% value difference so should have the same similarity).

3. **Linear or concave/convex?** - How does the similarity decrease as the attribute value gets further away from the target value?

4. **What is the minimum similarity?** Usually this is always zero if every item in the dataset has the specified attribute. When the attribute values are as far apart as possible (e.g. 60 and 80 for critical strike chance) the similarity should be minimal.

In our CBR system we represented range similarities by the following function where x is the target attribute value x_1 is the case attribute value and r is the specfied attribute range.

$$sim = \frac{r - |x - x_1|}{r}$$

The function satisfies symmetry, linearity and having minimum similarity of zero (when $x - x_1 = r$). These properties were selected as they provided a simple model. No good justifications could be found for deviating from them in this context.

Link/Socket Similarity - As specified above items in different link groups (≤ 4, 5 and 6) bear no similarity to each other. Therefore the similarity function returns 1 if items are in the same link or socket group and 0 otherwise.

Time Similarity - With time we generally assume that the target item is later than all items in the case base. However with system validation we need to be able to compare between items in the case base. Therefore a time similarity function should be symmetric. Given minutes m since the item sold (or in the case of validation the difference in minutes between each item selling) and some positive constant α, the local similarity can be defined as:

$$sim = \begin{cases} 0, & \text{if } 1 - \alpha m \leq 0 \\ 1 - \alpha m, & \text{if } 1 - \alpha m > 0 \end{cases}$$

α determines how at what time difference the similarity between two items becomes zero. The value of α depends heavily on the economic nature of the item. If an item fluctuates in price often then the value of α should be high. If an item maintains it's price over time then α can be lower.

3.4.3 Attribute Weights

For unique items relative importance of attributes is usually static over time as attributes tend to be closely interrelated (i.e. they all benefit the same type of character builds). This means we do not need to update the attribute weight distribution often. To justify initial weights we analyze statistical data.

From Table 3.2 it is clear that the number of item links is the primary determinant of the price of a Windripper.

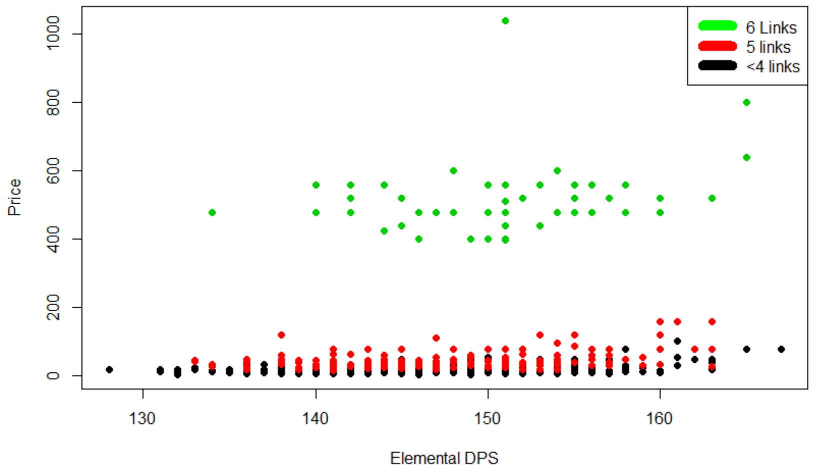


Figure 3.5: Windripper prices against Elemental DPS, coloured by link count.

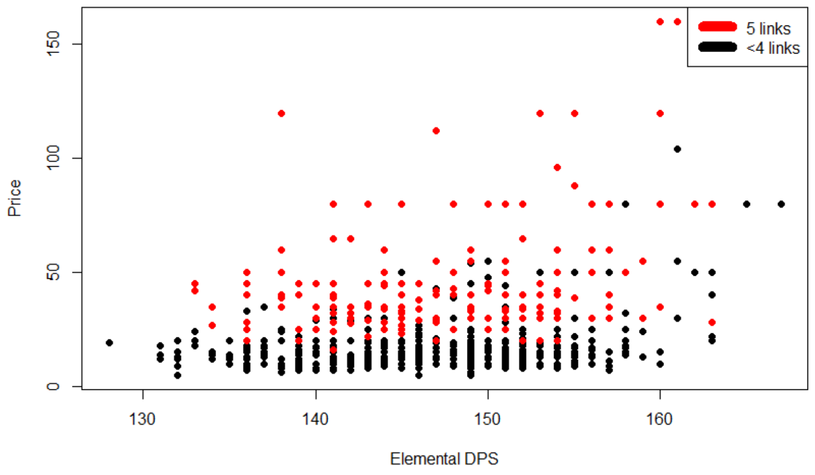


Figure 3.6: Zoomed in version of Figure 3.5 excluding 6 links.

Figure 3.5 and Figure 3.6 show the spread of 6 link, 5 link and 4 or less links Windrippers that have been sold in the pre-processed dataset. There is no overlap between 6 link Windrippers and 5 or less link Windrippers. Between 5 link and 4 or less link Windrippers there a small amount of overlap.

A Windripper with 6 links implies it has 6 sockets, however having 6 sockets does not imply 6 links. To determine the relative importance of sockets we need to control for the number of links. Figure 3.7 shows all sold Windrippers with 4 or less links, coloured by the number of sockets. Here we can see the distinction between socket groups is nowhere near as prominent as the distinction between link groups. This is evidence that link groups is more influential on the price of a Windripper than socket groups.

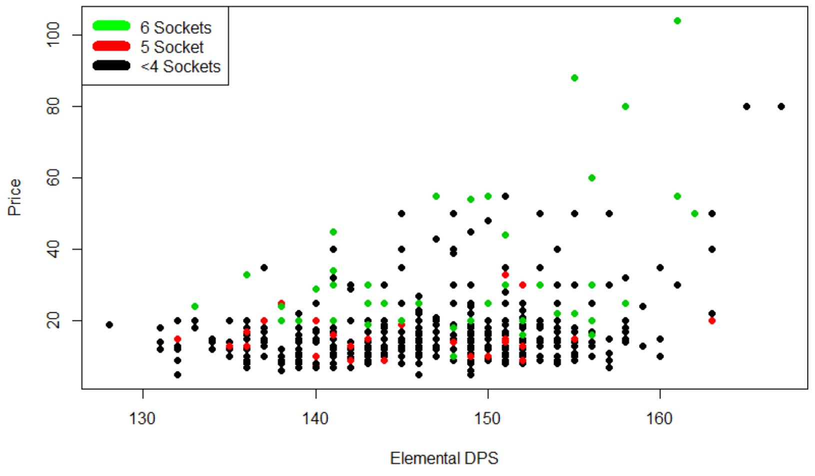


Figure 3.7: ≤ 4 linked Windrippers prices against Elemental DPS, coloured by socket count

To determine the relative importance of non-categorical variables (time, elemental dps and critical strike chance) we can use a multiple linear regression model. To control for links and sockets we excluded any items with 5 or more links or sockets. Firstly, we max-min normalized the attributes time, elemental dps and critical strike chance in statistical software package R using the following function:

```
normalize <- function(x) {
+     return ((x - min(x)) / (max(x) - min(x)))
+ }
```

Then we ran multiple linear regression with the price as the dependent variable and the above three normalized attributes as the explanatory variables. For $n = 553$ total Win-

Table 3.3: **Regression parameters for Windripper attributes on Windripper price**

Attribute	estimate	standard error	p
Crit Chance	11.00	1.34	0.00
Elemental DPS	19.48	2.31	0.00
Time Since Update	6.87	1.38	0.00

drippers, Table 3.3 shows the key results of the regression. All p-values were sufficiently close to zero which suggests that each of the above attributes has some meaningful impact on the price. Assuming a higher attribute is better, Elemental DPS is worth more than Critical Strike Chance as it has a higher estimator. The estimator for time since update tells us that these Windrippers generally increase in price over time (6.9c over the month period).

From this analysis we set our initial weights as ($w_{time} = 8, w_{eledps} = 6, w_{crit} = 5, w_{sock} =$

$30, w_{link} = 80$). From here we used the hill climbing approach specified previously to optimize weights. For simplicity we only considered weights at integer values. After optimization the final weights were ($w_{time} = 7, w_{eledps} = 19, w_{crit} = 11, w_{sock} = 30, w_{link} = 80$). Note because link and socket similarity are categorical, for weights of 10 or more the objective function did not change as the set of items returned in validation would always be the same. Relative percent error fell from 1.36293 for initial weights to 1.35795 for the optimized weights. This is a very marginal improvement which suggests that subjective methods to determine initial weights may not provide huge losses in predictive accuracy.

3.5 Rare Items

Rare items is where the pricing problem becomes most interesting. Whenever an item drops players need to determine whether it is worth selling or not. This is usually simple for commodities and unique items but is much harder for rare items where there is so many possible item combinations. Aside from having different attribute values, most rare items also have a different set of attributes. The probability two items have exactly the same set of attributes and attribute values is incredibly low. For example, a Diamond Ring rare item has at least 10^{16} possible combinations (see Appendix 1 for proof). To find similar items on the market using *poe.trade* players will usually filter by important attributes and attribute ranges. For example, a player wanting to find similar items to *Phoenix Hoof* in Figure 3.8 may apply the following search filter:

Maximum Life [60-80]
Total Elemental Resistance [70 - 90]
Movement Speed [20-30]

This filter assumes that evasion rating is not believed to be very important to the price of the item. From there players have to scan the results to determine what the approximate market price of the item is. This is time consuming and requires game knowledge of what mod values and combinations are valuable. We theorize that a CBR system will have the most benefit to users for heterogeneous items such as rares by quickly and (hopefully) accurately determining the fair market value of an item.

3.5.1 Datasets Analysis

Three datasets were sourced for rare items of base type Slink Boots, Rustic Sash and Diamond Ring. Multiple datasets were analysed as between base types varying levels of complexity exist. Slink Boots have a set of 15 affixes while Diamond Rings have a set of 35 affixes. Creating a CBR system that operates over several datasets allows us to generalize. Rare items are much more common than unique items so the datasets are much larger. This does not necessarily mean we will find more similar items in our case base as rare items are highly heterogeneous.

Table 3.4: **Sample statistics for Rare Items**

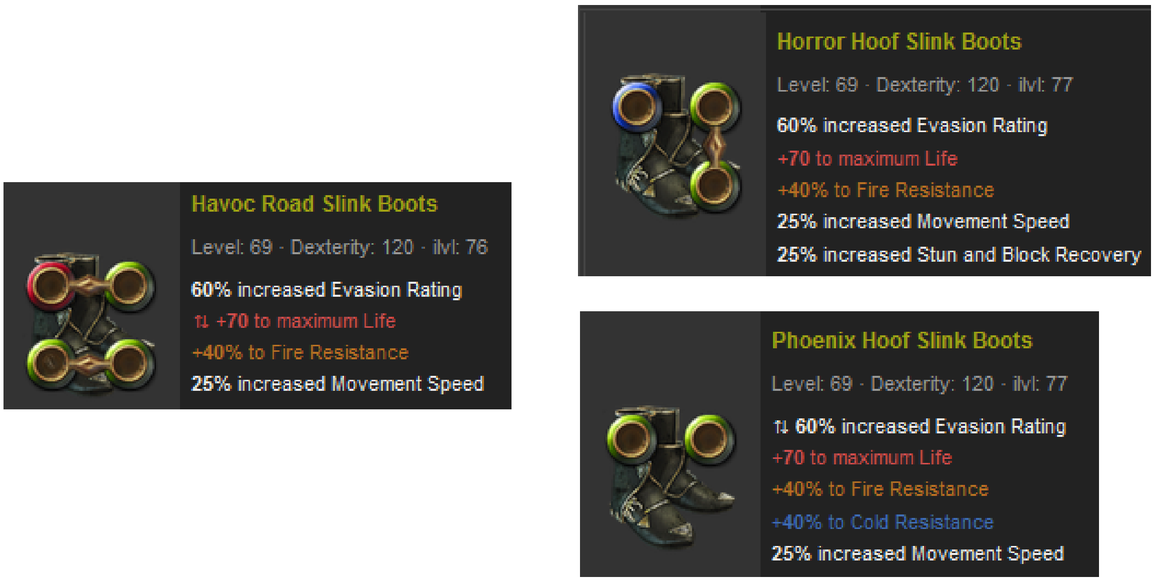
Rare Item	n	mean	SD	median
Slink Boots Sold	3383	5.90	22.05	1
Slink Boots On Market	2148	4.68	19.75	1.9325
Rustic Sash Sold	23122	5.88	27.29	1
Rustic Sash On Market	14493	5.02	18.98	1
Diamond Ring Sold	28130	8.70	56.99	2
Diamond Ring On Market	15704	7.99	28.44	2

From Table 3.4 we can see that rare items paint a contrasting picture to commodities and uniques. In all cases the average price and standard deviations of sold items is greater than those on the market. Also the proportion of items in the data set that are on the market is higher than uniques and commodities. A possible explanation is because rare items are more difficult to price, players may list rare items on the market at a low price in the hope that they will sell for something eventually. In reality most of these items are probably "garbage" so they never end up selling. Due to the heteroskadestic nature of the data, datasets with higher mean prices will have higher variances. This could explain why sold items tend to have larger variances.

3.5.2 Similarities

For local similarities every mod can be modeled by a range similarity function as specified in the unique item context. Because not every item has every attribute, the global similarity model used for unique items must be updated. A simple adaptation is that we can assume if an item does not have a particular attribute then the value of that attribute can be thought of as being equal to zero. This model works only if all items have the same number of attributes. If we compare items that share all of the same attributes and attribute values except one item has an extra attribute, such as in Figure 3.8, we can see where this fails. Under the proposed model with additive similarities *Havoc Road* will give the same similarity when compared with both *Horror Hoof* and *Phoenix Hoof.* However if Cold Resistance is a much more valuable attribute than Stun and Block Recovery then we would expect that the price difference between *Horror Hoof* and *Havoc Road* to be larger than *Phoenix Hoof* and *Havoc Road.* Therefore, *Horror Hoof* should be more similar to *Havoc Road* than *Phoenix Hoof.*

A solution is to implement negative similarities or dissimilarities. If two items do not share an attribute then we add a dissimilarity to the item. This dissimilarity is proportional to the weight of the attribute and also the value of the attribute. For



Havoc Road Slink Boots
Level: 69 · Dexterity: 120 · ilvl: 76
60% increased Evasion Rating
⇅ +70 to maximum Life
+40% to Fire Resistance
25% increased Movement Speed

Horror Hoof Slink Boots
Level: 69 · Dexterity: 120 · ilvl: 77
60% increased Evasion Rating
+70 to maximum Life
+40% to Fire Resistance
25% increased Movement Speed
25% increased Stun and Block Recovery

Phoenix Hoof Slink Boots
Level: 69 · Dexterity: 120 · ilvl: 77
⇅ 60% increased Evasion Rating
+70 to maximum Life
+40% to Fire Resistance
+40% to Cold Resistance
25% increased Movement Speed

Figure 3.8:

example in Figure 3.8, if *Phoenix Hoof* had a Cold Resistance of 5 it would be more similar to *Havoc Road* than having a Cold Resistance of 40. For mod i We can express dissimilarity by the following equation:

$$dissim = -\alpha n(\frac{w_i}{\sum_{i=0}^{n} w_i})(\frac{v(i)}{max(i)})$$

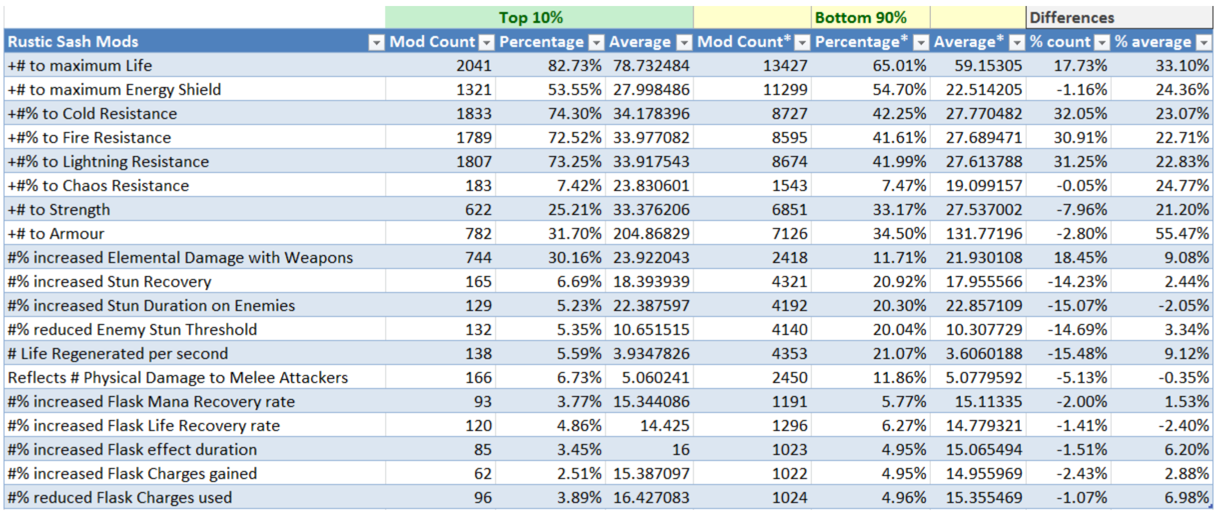
w_i is the weight of the mod and $\sum_{i=0}^{n} w_i$ is the sum of all mod weights. $v(i)$ is the value of the mod and $max(i)$ is the maximum value the mod can take. α is some positive constant which represents the weight we should put on excluded attributes. To determine the value of alpha we observed errors during validation (see evaluation section) across every item in the Slink Boots dataset. This was not possible for the Rustic Sash and Diamond Ring datasets, as the datasets were too large to run optimization (a single leave one out cross-validation takes over 4 hours). Because of this we assume the value of α generalizes to all rare item datasets. Through a simple hill-climbing approach we found the average absolute and relative error was minimized at $\alpha \approx 1.3$.

3.5.3 Attribute Weights

As opposed to unique items, rare items datasets are noisier, have higher dimensionality and are more likely to see the weight distribution change over time. Hill climbing with a with leave-one-out cross validation is no longer tractable. One solution is to use a less computationally expensive method such as 10x10 fold cross validation. However, these objective methods may end up fitting to the noise in the data.

Because of the high dimensionality and general noisiness of the data it was decided that rare items weights would be decided purely by subjective methods.

There are several statistical approaches we could consider to justify attribute weights. If a player has put an item up for sale and it has sold then that must mean at least some of the mods on that item gave it value. We could therefore look at the percentage of items sold with each mod. This only works if mods are uniformly distributed, however some mods are rarer than others - the exact percentages of mods appearing on an item are unknown within the game and depend on several factors.



	Top 10%			Bottom 90%			Differences	
Rustic Sash Mods	Mod Count	Percentage	Average	Mod Count*	Percentage*	Average*	% count	% average
+# to maximum Life	2041	82.73%	78.732484	13427	65.01%	59.15305	17.73%	33.10%
+# to maximum Energy Shield	1321	53.55%	27.998486	11299	54.70%	22.514205	-1.16%	24.36%
+#% to Cold Resistance	1833	74.30%	34.178396	8727	42.25%	27.770482	32.05%	23.07%
+#% to Fire Resistance	1789	72.52%	33.977082	8595	41.61%	27.689471	30.91%	22.71%
+#% to Lightning Resistance	1807	73.25%	33.917543	8674	41.99%	27.613788	31.25%	22.83%
+#% to Chaos Resistance	183	7.42%	23.830601	1543	7.47%	19.099157	-0.05%	24.77%
+# to Strength	622	25.21%	33.376206	6851	33.17%	27.537002	-7.96%	21.20%
+# to Armour	782	31.70%	204.86829	7126	34.50%	131.77196	-2.80%	55.47%
#% increased Elemental Damage with Weapons	744	30.16%	23.922043	2418	11.71%	21.930108	18.45%	9.08%
#% increased Stun Recovery	165	6.69%	18.393939	4321	20.92%	17.955566	-14.23%	2.44%
#% increased Stun Duration on Enemies	129	5.23%	22.387597	4192	20.30%	22.857109	-15.07%	-2.05%
#% reduced Enemy Stun Threshold	132	5.35%	10.651515	4140	20.04%	10.307729	-14.69%	3.34%
# Life Regenerated per second	138	5.59%	3.9347826	4353	21.07%	3.6060188	-15.48%	9.12%
Reflects # Physical Damage to Melee Attackers	166	6.73%	5.060241	2450	11.86%	5.0779592	-5.13%	-0.35%
#% increased Flask Mana Recovery rate	93	3.77%	15.344086	1191	5.77%	15.11335	-2.00%	1.53%
#% increased Flask Life Recovery rate	120	4.86%	14.425	1296	6.27%	14.779321	-1.41%	-2.40%
#% increased Flask effect duration	85	3.45%	16	1023	4.95%	15.065494	-1.51%	6.20%
#% increased Flask Charges gained	62	2.51%	15.387097	1022	4.95%	14.955969	-2.43%	2.88%
#% reduced Flask Charges used	96	3.89%	16.427083	1024	4.96%	15.355469	-1.07%	6.98%

Figure 3.9: Differences between attribute frequency and value for the top 10% and bottom 90% of Rustic Sash belts

Another observation is that sold rare items tend to follow a Zipfian distribution. That is few items are sold at a high price and many items are sold at a low price. We could observe which mods and mod values appear on valuable items compared to less valuable items. To do this we compared the top 10% of items sold by price to the bottom 90%. This partition occurs at a price of 10c for Rustic Sashes. Figure 3.9 displays the results for Rustic Sashes. Several mods such as Life, Elemental Resistances and Elemental Damage with Weapons appear to be valuable as the top 10% of Rustic Sashes have a significantly higher frequency and average value of these mods. Elemental Damage with Weapons seems to be a particularly desirable attribute with a 157.62% relative percentage increase in frequency. Most mods have a lower percentage frequency in the top 10% of items compared to the bottom 90%. This implies they are probably not very valuable. Inclusion of the mod probably means that it is occurring instead of a more valuable mod.

Some mods decrease in frequency but increase in average mod value such as Armour, Strength, Energy Shield and Chaos Resistance. I believe that this implies that these mods are desirable, but only in combination with other valuable mods. Where Life and Elemental Resistances are the primary determinants of a Rustic Sash' value the addition of the above mods with high mod values further add value to the item.

From this type of statistical analysis a rank order of mod importance could be created. Mod weights were arbitrarily determined. Appendix B displays the defined mod weights for the three different rare items. Optimization did not improve the accuracy of the CBR system for unique items by much. This suggests that subjective methods can produce ideal mod weightings for items. We have assumed that this observation applies also to rare items. This assumption may not be as valid for rare items due to the increased level of complexity.

3.6 Evaluation

3.6.1 Cross Validation

CBR systems can be internally evaluated through the use of cross validation. This involves partitioning the case base into a training set and a test set. Where the test set is tested on a case base consisting of the cases in the training set. If the test cases perform poorly on our training set then this could mean our CBR model is fundamentally flawed (incorrect weights and similarity functions), the case solutions themselves are poor or there is a lack of good coverage across the case base.

Leave One Out Cross Validation

The particular method we use for cross validation is "Leave-One-Out" Cross Validation (LOOCV). LOOCV is an extreme case of k-fold cross validation where k is set to n, the number of items in the case base. This means we have n passes over the case base, with training sets of size $n - 1$ and test sets of size 1. Effectively, every case within the case base is being evaluated against every other case.

Validation Goals

Cross validation has a number of uses within our CBR system. Firstly, it gives us an alternative method of detecting outliers. Every item in the case base is compared against it's nearest neighbors. If the CBR system returns a solution (price) that is vastly different from the case's solution (price) then this case is likely an outlier which should be removed from the case base.

Cross Validation is also useful for optimization of parameters including attribute weights and the number of nearest neighbours k that have maximum similarity to the target case. By systematically adjusting the parameters and running cross validation over the entire case base at each step we can observe where the average case error is minimized. Parameter optimization is computationally expensive as each pass over the case base requires $(n - 1)$ similarity calculations over a set of m attributes. Because of this opt for local optimization techniques where we estimate realistic initial parameters and then vary the parameter by a fixed interval until we reach some locally optimal solution.

k - optimization

KNN finds the k most similar neighbours to the target case, then returns the solution (price) as a function of the solutions to these similar cases. The KNN algorithm can be optimized for each dataset by varying k, the number of neighbors returned.

If the value of k is too high then the CBR system returns too many unrelated items. If the value of k is set too low then the CBR system lacks sufficient cases to make an informed decision [19], especially if there is a lot of noise in the case solutions. Poliscastro et al. [18] noted the optimal value of k is determinant on similarity characteristics specific to each problem. Therefore, different data sets will have different optimal values of k. k is determined in our system by trial and error cross-validation. Figure 3.10 shows how

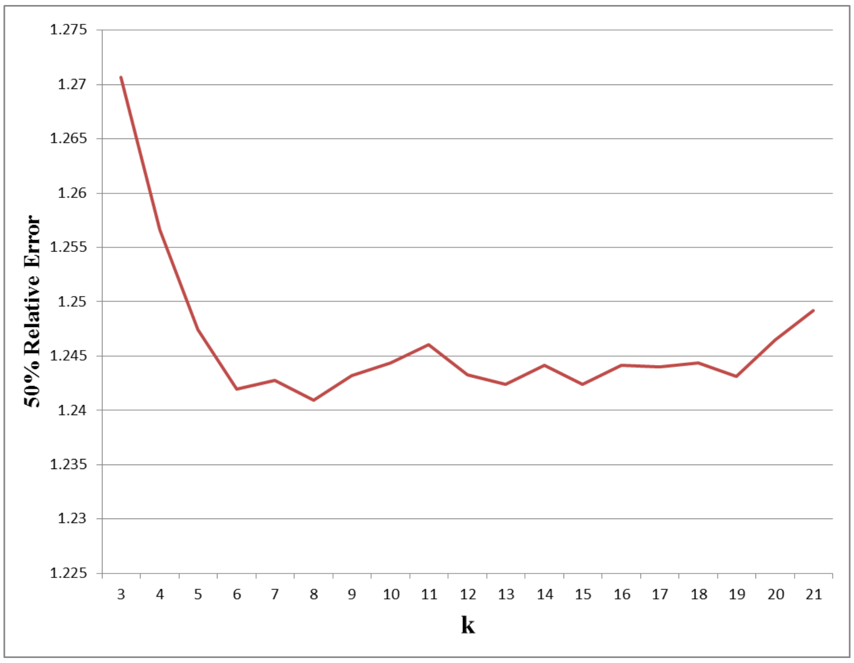


Figure 3.10: k optimization for the Windripper dataset

the average error changes as k increases. Generally, as k increases the error decreases quickly then slowly tracks back upwards. For the Windripper dataset k was optimized at 8. For Lioneye's fall it was optimized at 3 and for Slink Boots it was optimized at 12. The Diamond Ring and Rustic Sash datasets were too large to perform this optimization so we assumed $k = 12$, the same as the Slink Boots dataset. This assumes that rare datasets share the same intrinsic properties.

A possible reason why k is optimized at higher values for rare items compared to uniques and commodities is that the datasets are more noisy, so by returning more items we are helping smooth that noise.

Error

When optimizing using cross validation we look to find the minimizing average error. However this error can be defined in different ways. Considering two prices p_1 and p_2.

1. **Absolute Error** - the absolute difference between two prices $|p_1 - p_2|$
2. **Relative Error** - the ratio between the two prices $max(\frac{p_1}{p_2}, \frac{p_2}{p_1})$

For example, a 6 linked Windripper has an average price of 627c while a non-linked Windripper has an average price of 18.5c. If our CBR model predicted the price of a 6 linked Windripper at 627c but is really sold for 550c the absolute error would be 77 while the relative error would be 1.14. If our CBR model predicted a non-linked Windripper at 18.5c but it really sold for 10c then the absolute error would we 8.5 while the relative error would be 1.85. Depending on which metric we look at we get contradictory results. Relative errors provide better representations for heteroskedastic data where the variance in prices increases as the item becomes more valuable. However at very low item prices, relative errors can be misleading. Take for instance a rare item that is predicted at 2c but sold for 0.125c and another item that is predicted at 160c but sold for 10c. Although in both instances they give the same relative error of 16 the latter case is far more problematic in terms of predictive accuracy.

Because errors paint a different picture based upon the nature of the data it is difficult to comparatively evaluate CBR systems between different item types. However, for optimization and outlier detection purposes targeting minimizing the average error is still a meaningful objective. To help alleviate the weaknesses of the above metrics we consider a 50% mean trimmed error. This is the mean of the error values between the 25% lower and 75% upper quartile. This gives a better picture of a typical error for the dataset by excluding abnormal errors. For cross validation both the 50% trimmed absolute and relative errors are both calculated, although the objective function aims to minimize the 50% trimmed relative error. In practice the 50% trimmed mean relative error is around half of the average relative error over all items for rare datasets.

Cross-Validation Findings

Table 3.5 shows that commodities and unique items have significantly lower relative errors than rare items. This is expected due to the homogeneity of these items many similar items can be found. Also the variance between prices of these items is far less than for rares.

Table 3.5: **Optimized LOOCV average errors for sold items in each dataset**

Dataset	50% Rel Error	50% Abs Error	No of Attributes	Mean Price
Lioneye's Fall	1.078	1.063	1	14.87
Windripper	1.241	4.567	5	55.40
Slink Boots	1.640	0.854	15	5.900
Rustic Sash	1.587	0.730	20	5.882
Diamond Ring	1.852	1.269	35	8.698

Items with a higher mean price have larger absolute errors. This observation fits with the idea that item data is heteroskaedestic, so item variance increases as price increases.

Within the rare item datasets Slink Boots and Rustic Sash' have similar mean prices, number of attributes as well as errors. The Diamond Ring dataset has a significantly larger relative error. This could be due to the high dimensionality of the data meaning there is a larger variety of different items on the market making finding highly similar items more difficult. However, a single data point is not enough to draw this conclusion.

3.6.2 External Evaluation

Internal validation may not tell us much when the data is noisy. As we have investigated the quality of the solution (price) for cases (items) within the case base can be questionable. Another way we could potentially evaluate our model is through some sort of external validation.

Expert Agreement

We could give an *expert* a set of items to calculate what they believe is a fair market price for each item. As described earlier, they would likely use their prior trading experience with similar items as well as comparing similar items listed on the market to formulate their valuation. We could then use our CBR system with the same set of items to return a price for each item. If there is general agreement between the expert's prices and the CBR system's prices then that would add validity to our CBR model. The key assumption with this method of valuation is that "experts" can themselves price items accurately, which may not be true as humans usually have limited ability to consider all the determinant variables, instead relying on heuristic methods.

Trading Items

A practical way we could evaluate the system is by buying up a set of items on the market (regardless of their price), determining their price using the CBR system then re-listing them on the market at that price. There are two cases we have to consider:

- **Under-priced** - If you get bombarded by trade requests for the item soon after listing it on the market it is a fair indication you have under-priced the item.
- **Over-Priced** - The item does not sell after a long period of time (say about a week). Note at fixed intervals (say each day) the CBR system should re-determine and update the price of the item.

The problem with under-priced items is we don't have any indication of how under-priced the item was. If the item sold for 5c would it also have sold at 10c? What about at 50c? Also the item selling quickly may not indicate necessarily that the item was under-priced. It could be a buyer was searching for the item at the right time. Because of the temporal nature of the validation as well as not being able to efficiently process a large sample of items this evaluation method may not be well suited for refining a CBR pricing system.

A variation on this method is scanning a list of items on the market and buying any item that is under-priced according to the CBR system (by some threshold value). These items could then be re-listed at the CBR determined price. This is a computational approach to *item flipping*. The amount of profit that is made in a given timeframe could be used to evaluate the effectiveness of the CBR model. Note that some item markets will be more efficient than others (i.e commodities over rares) so evaluation of different models must take place using similar items. Using the same items for sequential evaluation of different models should also be avoided. If the first evaluation detects the majority of under-priced items the second evaluation (assuming it comes soon after) will be starved of profit opportunities, meaning the two evaluations are not independent.

4 Discussion

4.0.1 Limitations

Sourcing Market Data

The Exile-Tools API was shut down during development of the case based system due to high server costs. Initially the intent of the system was to be able to source live market data. Building an item indexer from scratch would have been too time consuming and requires analyzing a data stream of several terabytes per month. Before the indexer shut down multiple datasets of historical data for certain items was downloaded. These datasets were aimed to be representative of the different item categories in the *Path of Exile* economy.

The nature of the indexer means that we cannot know if an item actually sold for its listed price, only that it was removed from the market at that price. This means there is no guarantee on the quality of the solutions in our case base. Due to the difficulty in pricing rare items, some players may list several rare items on the market in the hope that they sell for some small amount. If they do not sell over a period of time then the player will take the items off the market to be discarded. The indexer will consider these items as being sold.

Item Assumptions

Although we tried to model all the attributes that determine the price of an item, there are some notable exclusions that are hard to model. A rare item can have a maximum of 6 mods. An Exalted Orb can be used on an item to add an extra random mod. Master crafting can also be used to add an additional "crafted" mod to an item. In this case often an item with 5 mods may be superior to an item with the same 5 mods plus a bad mod. Modeling this exclusion is difficult as depending on whether the mods are prefixes or suffixes determines what mods can be crafted on the item. This breaks the assumption that more mods is always better.

Certain items can also be enchanted or corrupted giving them additional valuable mods. The number of possible corruptions and enchantments is very large which would greatly increase the dimensionality of the system. Because of this items that had these additional mods were excluded from the case base. In all datasets these items made up a very small fraction ($< 1\%$) of the total transactions.

Market Manipulation

A symptom of sellers not having to honour the listed price of an item is the increasing prevalence of "fake listings" [7]. These sellers list items at well below market price with no intention of selling them. The intention is that other players will list their items at the same price as the fake listings, at which point the price manipulator can buy up the item for well under it's true value. Other forms of manipulation are possible such as price fixing a small market (for example a niche unique item) but are harder to detect. Blatant market manipulation is frowned upon by the community but is not regulated in any way by the developers. 3rd party tools have been developed for players to blacklist sellers that manipulate prices [3]. While our CBR system does not take into account items that have not sold on the market, price manipulation increases the chance of outliers appearing in the data, which adversely affects the accuracy of the system.

Price Shocks

A price shock is an unexpected event that drastically changes the price of an item due to exogenous factors. For example, on 25/9/16 a popular community member Ghudda released a video highlighting an extremely powerful but not publicly known build that centered around the commodity item *Essence of Delirium* [4]. Within a few hours the price of *Essence of Delirium* had more than tripled. Such shocks are hard to account for in our system if the time similarity attribute is not weighted highly. By returning items that have not sold very recently, a price shock means pricing data is going to be inaccurate. This is not a problem for commodity items which consider the price of the last k items sold.

A potential solution could be to have an adaptive time similarity weight. If average prices start deviating more than usual then we could increase the time similarity weight so the predicted CBR price is more likely to reflect recently sold items. This would require implementing some sort of change detector.

4.0.2 Conclusion

At the lowest level Economics is concerned with how humans make decisions. Often it is assumed that agents are perfectly rational creatures that make optimal decisions in all scenarios. However we know this is never true as humans always have limited time and knowledge available. We have observed in the virtual economy of the MMO *Path of Exile* that players often make constant decisions on whether in game items are worth selling to other players and if so how much to sell them for? This "pricing problem" results in players employing a range of pricing strategies to efficiently price items given time and knowledge constraints.

We have characterized the different types of items available in *Path of Exile*, observing that heterogeneous items (rare items) are more difficult to price than homogeneous items (commodities and uniques) resulting in inefficient markets. To address the "pricing problem" we designed and implemented a novel CBR system to determine the fair market price of an un-priced item.

This dissertation extensively describes the design considerations when building the CBR pricing system. Rare items are more complex and have higher dimensionality than commodities and unique items resulting in more complex models for similarity between items. We found when internally evaluating the system using cross-validation that rare items give larger errors. This can be attributed to the fact that the CBR system infers from historical transactions. Market inefficiency means that many items sell at prices higher or lower than the fair market price. This results in "noisy" data where similar items may sell at a range of different prices.

A CBR approach has been found to be effective for pricing items in this domain. Virtual economies are becoming increasingly hard to ignore, with some contributing more GDP than many countries [1]. The findings from this dissertation could be used by practitioners to generalize to other emerging virtual economies with similar characteristics.

4.0.3 Future work

In future we would hopefully create or have access to a new item indexer that tracks items on the market and stores historical data. This would allow for the real-time ability to price items on the market. It could also be used to trawl the market for items that have been under-priced to flip them for profit.

To make our system more generalizable we should consider methods for automatic weight detection [16]. It is not feasible for developers to set the weights for every item on the market. Similarly we should define a method for automatic outlier detection and removal using the evaluation system (for example removing any item with a LOOCV error greater than some threshold).

Bibliography

[1] Thorpe, Christopher, et al. Virtual economies: threats and risks. International Conference on Financial Cryptography and Data Security. Springer Berlin Heidelberg, (2007).

[2] Castronova, Edward. On virtual economies. (2002).

[3] MisaMisa. "poe.trade - Blacklist Script" www.pathofexile.com/forum/view-thread/1741446

[4] Ghudda. "Speedy Decay Build". https://www.pathofexile.com/forum/view-thread/1742921

[5] Thirdy. "Durian". https://www.pathofexile.com/forum/view-thread/1507190

[6] Rasmuki. "PoE ninja". http://poe.ninja/esc/currency

[7] I_NO. "Market manipulation". https://www.pathofexile.com/forum/view-thread/1652270

[8] acme_myst. "About Item Flipping and Path of Economy" https://www.pathofexile.com/forum/view-thread/1274923/page/1

[9] Waterman, Pete. "ExileTools Indexer". https://github.com/trackpete/exiletools-indexer

[10] Kreul, Lee M. Magic numbers: Psychological aspects of menu pricing. Cornell Hotel and Restaurant Administration Quarterly 23.2 (1982): 70-75.

[11] Kolodner JL. An introduction to case-based reasoning. Artificial Intelligence Review (1992).

[12] Watson, Ian. Case-based reasoning is a methodology not a technology. Knowledge-based systems 12.5 (1999): 303-308.

[13] Kim GH, An SH, Kang KI. Comparison of construction cost estimating models based on regression analysis, neural networks, and case-based reasoning. Building and Environment (2004): 39(10): 1235-42.

[14] Arditi D, Tokdemir OB. Using case-based reasoning to predict the outcome of construction litigation. Computer-Aided Civil and Infrastructure Engineering (1999): 14(6):385-93.

[15] An, Sung-Hoon, Gwang-Hee Kim, and Kyung-In Kang. A case-based reasoning cost estimating model using experience by analytic hierarchy process. Building and Environment 42.7 (2007): 2573-2579.

[16] Yan, Aijun, Hongshan Shao, and Zhen Guo. Weight optimization for case-based reasoning using membrane computing. Information Sciences 287 (2014): 109-120.

[17] Lin, Shih-Wei, and Shih-Chieh Chen. Parameter tuning, feature selection and weight assignment of features for case-based reasoning by artificial immune system. Applied Soft Computing 11.8 (2011): 5042-5052.

[18] C.A. Policastro, C.P.L.F. Carvalho André, C.B. Delbem Alexandre, A hybrid case adaptation approach for case based reasoning, Applied Intelligence 28 (2008) 101-119.

[19] C.-S. Park, I. Han, A case based reasoning with the feature weights derived by analytic hierarchy process for bankruptcy prediction, Expert Systems with Applications 23 (2002) 255-264.

[20] K.D. Althoff, Knowledge acquisition in the domain of CNC machine centers the MOLTKE approach, in: J. Boose, B. Gaines, J.G. Ganascia, (Eds.), Proceedings of Third European Workshop on Knowledge-Based Systems, (1989).

[21] Kolodner, Janet. Case-based reasoning. Morgan Kaufmann, (2014).

[22] Richter, Michael M., and Rosina O. Weber. Case-based reasoning. A Textbook. 546 (2013).

[23] Gayer, Gabrielle, Itzhak Gilboa, and Offer Lieberman. Rule-based and case-based reasoning in housing prices. The BE Journal of Theoretical Economics 7.1 (2007).

[24] Musa, Adebola G., et al. A Neural-CBR System for Real Property Valuation. Journal of Emerging Trends in Computing and Information Sciences 4.8 (2013): 611-622.

[25] Heeks, Richard. Understanding gold farming and real-money trading as the intersection of real and virtual economies. Journal For Virtual Worlds Research 2.4 (2009).

[26] Prax, Patrick. The Commodification of Play in Diablo 3. Understanding the Real Money Market Place. (2012).

[27] Prax, Patrick. Game Design and Business Model: an Analysis of Diablo 3. Proceedings of DiGRA. (2013).

[28] Salter, Alexander William, and Solomon Stein. Endogenous currency formation in an online environment: The case of Diablo II. The Review of Austrian Economics 29.1 (2016): 53-66.

[29] Rossini PA, Accuracy Issues for Automated and Artificial Intelligent Residential Valuation Systems, International Real Estate Society Conference, Kuala Lumpur, January 26-30. (1999)

[30] Skitmore RM, Thomas NS. Forecast models for actual construction time and cost. Building and Environment (2003): 38(8):1075-83.

[31] Nawawi AH, Jenkins D and Gronow S Expert system development for the mass appraisal of commercial property in Malaysia. Journal of the Society of Surveying Technicians 18(8): 66-72. (1997)

[32] Almond N, Lewis O, Jenkins D, Gronow S and Ware A ?Intelligent systems for the valuation of residential property?. RICS Cutting Edge, Conference, Dublin 5-6 September. (1997)

[33] Bode J. Neural networks for cost estimation. Cost Engineering 1998: 40(1):25-30.

[34] Alexa. “poe.trade Traffic statistics” http://www.alexa.com/siteinfo/poe.trade

Appendix

.1 Diamond Ring Combinatorics

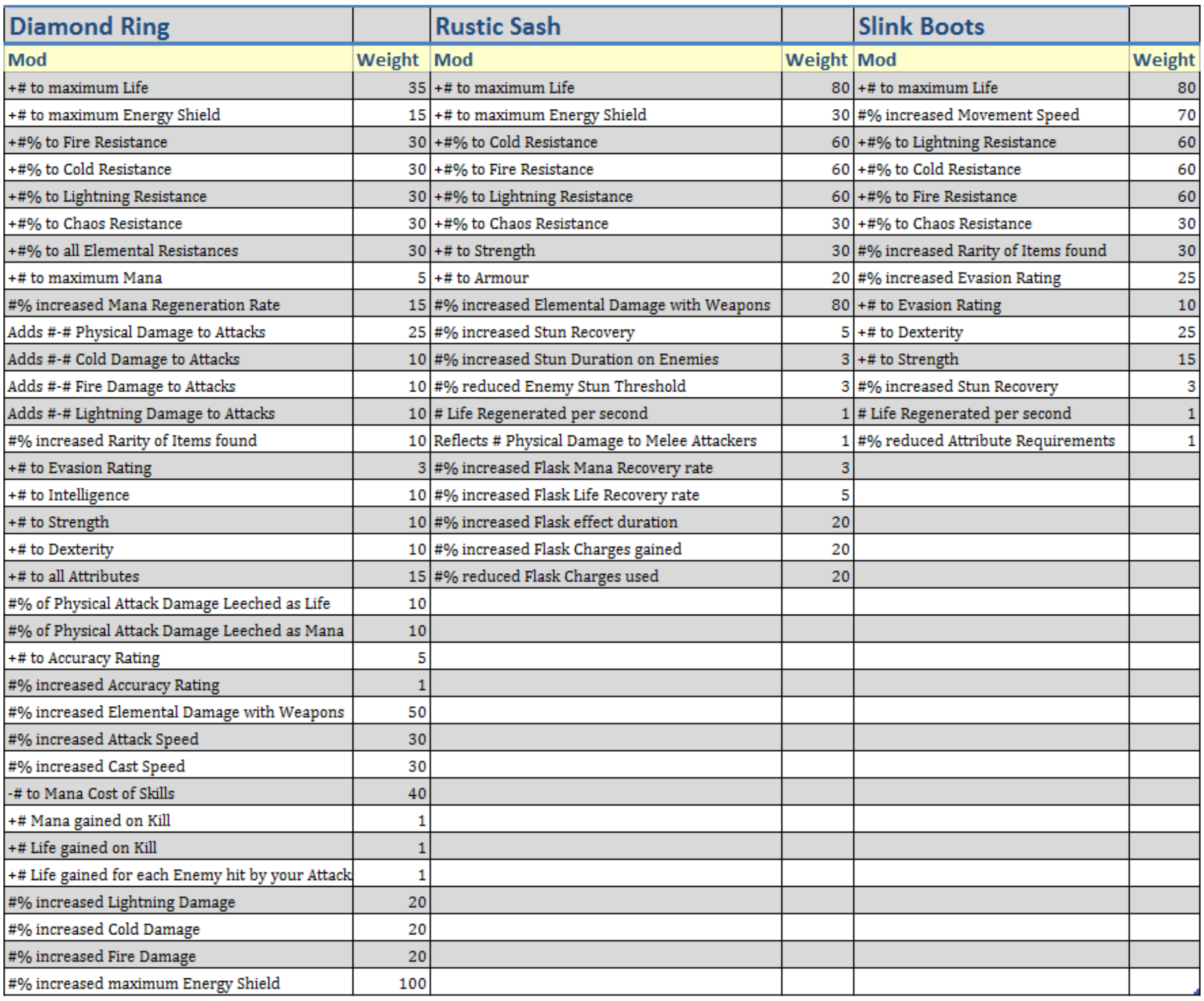
A rare Diamond ring has 13 available prefixes and 22 available suffixes. If we assume that each mod has an equally likely chance to appear (some mods are rarer than others). And that each mod has on average 40 numerical values it can take (obtained by taking the average of the sum of all mod ranges for diamond rings). Also that rare items have between 4-6 mods. The list of possible mod combinations are:

$$\{pppsssm, ppsss, psss, pppss, ppps\}$$

$$count(pppssm) = {}^{13}C_3 \times {}^{22}C_3 \times 40^6 =$$

Which evaluates to approximately 1.8×10^{16}. The other combinations are of a lower order so will not affect the magnitude of the total count significantly. Therefore we can use this number as an approximate lower bound of the number of possible Diamond rings.

.2 Rare Item Mod Weights



Diamond Ring		Rustic Sash		Slink Boots	
Mod	Weight	Mod	Weight	Mod	Weight
+# to maximum Life	35	+# to maximum Life	80	+# to maximum Life	80
+# to maximum Energy Shield	15	+# to maximum Energy Shield	30	#% increased Movement Speed	70
+#% to Fire Resistance	30	+#% to Cold Resistance	60	+#% to Lightning Resistance	60
+#% to Cold Resistance	30	+#% to Fire Resistance	60	+#% to Cold Resistance	60
+#% to Lightning Resistance	30	+#% to Lightning Resistance	60	+#% to Fire Resistance	60
+#% to Chaos Resistance	30	+#% to Chaos Resistance	30	+#% to Chaos Resistance	30
+#% to all Elemental Resistances	30	+# to Strength	30	#% increased Rarity of Items found	30
+# to maximum Mana	5	+# to Armour	20	#% increased Evasion Rating	25
#% increased Mana Regeneration Rate	15	#% increased Elemental Damage with Weapons	80	+# to Evasion Rating	10
Adds #-# Physical Damage to Attacks	25	#% increased Stun Recovery	5	+# to Dexterity	25
Adds #-# Cold Damage to Attacks	10	#% increased Stun Duration on Enemies	3	+# to Strength	15
Adds #-# Fire Damage to Attacks	10	#% reduced Enemy Stun Threshold	3	#% increased Stun Recovery	3
Adds #-# Lightning Damage to Attacks	10	# Life Regenerated per second	1	# Life Regenerated per second	1
#% increased Rarity of Items found	10	Reflects # Physical Damage to Melee Attackers	1	#% reduced Attribute Requirements	1
+# to Evasion Rating	3	#% increased Flask Mana Recovery rate	3		
+# to Intelligence	10	#% increased Flask Life Recovery rate	5		
+# to Strength	10	#% increased Flask effect duration	20		
+# to Dexterity	10	#% increased Flask Charges gained	20		
+# to all Attributes	15	#% reduced Flask Charges used	20		
#% of Physical Attack Damage Leeched as Life	10				
#% of Physical Attack Damage Leeched as Mana	10				
+# to Accuracy Rating	5				
#% increased Accuracy Rating	1				
#% increased Elemental Damage with Weapons	50				
#% increased Attack Speed	30				
#% increased Cast Speed	30				
-# to Mana Cost of Skills	40				
+# Mana gained on Kill	1				
+# Life gained on Kill	1				
+# Life gained for each Enemy hit by your Attack	1				
#% increased Lightning Damage	20				
#% increased Cold Damage	20				
#% increased Fire Damage	20				
#% increased maximum Energy Shield	100				

.3 Example Item Data

```
{
 "_index" : "poe20160607",
  "_type" : "item",
  "_id" :
      "d513caa4981488b4d28122feedd39bc2401c58d323ee607cc98b9ba1e7edc3d1",
  "_score" : 9.793697,
  "_source" : {
    "info" : {
      "icon" :
          "http://webcdn.pathofexile.com/image/Art/2DItems/Weapons/TwoHandWeapons/Bows/
      "fullName" : "Windripper Imperial Bow",
      "flavourText" : "It hunts; as silent as falling snow, as deadly as
          the tempest.",
      "name" : "Windripper",
      "typeLine" : "Imperial Bow",
      "tokenized" : {
        "fullName" : "windripper imperial bow",
        "flavourText" : "it hunts; as silent as falling snow, as deadly as
            the tempest."
      }
    },
    "requirements" : {
      "Level" : 66,
      "Dex" : 212
    },
    "propertiesPseudo" : {
      "Weapon" : {
        "estimatedQ20" : {
          "Physical DPS" : 92,
          "Total DPS" : 241
        }
      }
    },
    "uuid" :
        "d513caa4981488b4d28122feedd39bc2401c58d323ee607cc98b9ba1e7edc3d1",
    "modsTotal" : {
      "#% increased Elemental Damage with Weapons" : 7,
      "Adds #-# Lightning Damage" : {
        "min" : 1,
        "avg" : 50.5,
        "max" : 100
      },
      "#% increased Attack Speed" : 12,
      "#% increased Quantity of Items Dropped by Slain Frozen Enemies" : 15,
      "Adds #-# Cold Damage" : {
        "min" : 32,
        "avg" : 41.5,
        "max" : 51
      },
      "#% increased Critical Strike Chance" : 70,
      "#% increased Rarity of Items Dropped by Slain Shocked Enemies" : 30
    },
    "properties" : {
      "Weapon" : {
        "Cold Damage" : {
          "min" : 32,
          "avg" : 41,
          "max" : 51
        },
        "Critical Strike Chance" : "8.5",
        "Lightning DPS" : 81,
        "Total Damage" : {
          "min" : 52,
          "avg" : 140,
          "max" : 229
        },
        "Cold DPS" : 66,
        "Elemental DPS" : 149,
        "Attacks per Second" : "1.62",
        "Physical DPS" : 77,
        "Physical Damage" : {
          "min" : 19,
          "avg" : 48,
          "max" : 78
        },
        "Total DPS" : 226,
        "type" : {
          "Bow" : true
        },
        "Lightning Damage" : {
          "min" : 1,
          "avg" : 50,
          "max" : 100
        },
        "Elemental Damage" : {
          "min" : 33,
          "avg" : 92,
          "max" : 151
        }
      }
    },
    "shop" : {
      "added" : 1465337343000,
    "chaosEquiv" : 3,
    "currency" : "Chaos Orb",
    "defaultMessage" : "@CaityRunsWithShields I would like to buy your
       Windripper Imperial Bow listed for 3 Chaos Orb (League:Prophecy,
       Stash Tab:\"Shop\" [x7,y6])",
    "lastCharacterName" : "CaityRunsWithShields",
    "hasPrice" : true,
    "verified" : "GONE",
    "updated" : 1465337845000,
    "priceSource" : "note",
    "stash" : {
      "inventoryID" : "Stash6",
      "xLocation" : "7",
      "yLocation" : "6",
      "stashID" :
         "f2b19c736d1b9ca291c6cea769f95d7e60ebdae3d730d5695f12bba5868de7ef",
      "stashName" : "Shop"
    },
    "note" : "~b/o 3 chaos",
    "modified" : 1465337845000,
    "saleType" : "b/o",
    "shelfLife" : 517,
    "amount" : "3",
    "sellerAccount" : "TheLittleFox",
    "price" : {
      "Chaos Orb" : 3
    }
  },
  "sockets" : {
    "totalGreen" : 2,
    "allSocketsGGG" : "G-B G",
    "sortedLinkGroup" : {
      "1" : "G",
      "0" : "BG"
    },
    "allSockets" : "GB-G",
    "totalBlue" : 1,
    "allSocketsSorted" : "BGG",
    "socketCount" : 3,
    "largestLinkGroup" : 2
  },
"mods" : {
  "Bow" : {
    "explicit" : {
      "Adds #-# Lightning Damage" : {
        "min" : 1,
        "avg" : 50.5,
        "max" : 100
      },
      "#% increased Attack Speed" : 12,
      "#% increased Quantity of Items Dropped by Slain Frozen Enemies"
          : 15,
      "Adds #-# Cold Damage" : {
        "min" : 32,
        "avg" : 41.5,
        "max" : 51
      },
      "#% increased Critical Strike Chance" : 70,
      "#% increased Rarity of Items Dropped by Slain Shocked Enemies" :
          30
    },
    "implicit" : {
      "#% increased Elemental Damage with Weapons" : 7
    }
  }
},
"attributes" : {
  "equipType" : "Bow",
  "weaponType" : "Bow",
  "corrupted" : false,
  "explicitModsCount" : 6,
  "baseItemName" : "Imperial Bow",
  "lockedToCharacter" : false,
  "identified" : true,
  "ilvl" : 66,
  "baseItemType" : "Weapon",
  "inventoryHeight" : 4,
  "rarity" : "Unique",
  "mirrored" : false,
  "implicitModsCount" : 1,
  "frameType" : 3,
  "league" : "Prophecy",
        "itemType" : "Bow",
        "inventoryWidth" : 2
      }
    }
  }
```