QUNBAR, SA'ED ALI, Ph.D. Automated Item Difficulty Modeling with Test Item Representations. (2019)
Directed by Dr. John T. Willse. 108 pp.

This work presents a study that used distributed language representations of test items to model test item difficulty. Distributed language representations are low-dimensional numeric representations of written language inspired and generated by artificial neural network architecture. The research begins with a discussion of the importance of item difficulty modeling in the context of psychometric measurement. A review of the literature synthesizes the most recent automated approaches to item difficulty modeling, introduces distributed language representations, and presents relevant predictive modeling methods. The present study used an item bank from a certification examination in a scientific field as its data set. The study first generated and assessed the quality of distributed item representations with a multi-class similarity comparison. Then, the distributed item representations were used to train and test predictive models. The multi-class similarity task showed that the distributed representations of items were more similar on average to items within their content domain versus outside of their domain in 14 out of 25 domains. The prediction task did not produce any meaningful predictions from the distributed representations. The study ends with a discussion of limitations and potential avenues for future research.

AUTOMATED ITEM DIFFICULTY MODELING WITH TEST ITEM REPRESENTATIONS

by

Sa'ed Ali Qunbar

A Dissertation Submitted to
the Faculty of The Graduate School at
The University of North Carolina at Greensboro
in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Greensboro
2019

Approved by

_____
Committee Chair

APPROVAL PAGE

This dissertation written by SA'ED ALI QUNBAR has been approved by the following committee of the Faculty of The Graduate School at The University of North Carolina at Greensboro.

Committee Chair _____
Dr. John Willse

Committee Members _____
Dr. Arnie Aldridge

_____
Dr. Micheline Chaloub-Deville

_____
Dr. Robert Henson

_____
Date of Acceptance by Committee

_____
Date of Final Oral Examination

ACKNOWLEDGEMENTS

I'd like to thank a few of the incredible individuals who have provided tremendous support over the years. My advisor and committee chair Dr. John Willse for being generous with straightforward advice in each of our countless meetings over the years. The members of my committee Dr. Arnie Aldridge, Dr. Micheline Chaloub-Deville, and Dr. Robert Henson for pushing me to open-minded, yet rigorous. The staff of the American Board of Pediatrics, especially Dr. Linda Althouse, Dr. Andy Dwyer, and Dr. Robert Furter, for their support of this project and my professional development. The faculty of the Educational Research Methodology department for bestowing (or at least attempting to bestow) immense knowledge onto me during my time as a student in the department. Dr. Thomas Kwapil for giving my first position as an undergraduate researcher and being the first to show me what it was to perform research at the highest level. Dr. Paul Silvia for very influential direct and all manner indirect support. Dr. Stuart Marcovitch who taught my first statistics class in a way that fostered the fascination and curiosity for the subject I still have today. My parents, siblings, friends, and colleagues who are always liberal with their encouragement and whose absolute certainty that I will achieve my goals often eclipses my own.

This project was supported in part by the American Board of Pediatrics. The content is solely the responsibility of the authors and does not necessarily represent the official views of the American Board of Pediatrics or the American Board of Pediatrics Foundation.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

Automatic item generation systems (AIG, Gierl & Haladyna, 2013; Gierl & Lai, 2016) and test development paradigms (Embretson, 1998; assessment engineering, Luecht, 2013; evidence-centered design, Mislevy, Almond, & Lukas, 2003) are welcomed innovations for item-hungry large-scale assessment programs, especially those utilizing computer adaptive testing (CAT) systems. Unfortunately, improvements in item pilot testing (item field-testing, or item piloting) have yet to bridge the gap between the recent advancements in item conceptualization, generation, and delivery. Item piloting is the process of collecting statistical information on an item by administering it on one or more live assessments. Item piloting remains a resource bottleneck in most testing frameworks. Whether intended for use in CAT systems or traditional delivery, the utility of large pools of automatically generated items cannot be fully realized if item piloting techniques do not catch up to the efficiency levels of the techniques used to develop those pools.

The Costs of Item Pilot Testing

Identifying the statistical properties of developmental items relative to the target population is the critical goal of item piloting within testing programs that utilize an item response theory (IRT) modeling framework. However, pilot testing items is a resource

intensive process. The costs incurred by testing organizations to collect item statistics through piloting can be great.

Time. Including an item on a test form before it can be used operationally adds time to the developmental lifespan of the item. The opportunity cost of selecting one item to pilot test instead of others is a similar issue given the limited opportunities for pilot testing. Selecting an item for pilot testing that ultimately performs poorly leaves test developers in a situation in which they must choose between possibly better performing, yet un-piloted items, or reuse existing items. Reusing existing items introduces the next cost of item pilot testing.

Exposure. Leaning on previously used statistically sound items increases the exposure of those items to the target population. Increasing exposure means increasing the chance that items are compromised. Compromised items undermine the validity of the assessment instrument. The risk of item compromise due to exposure also applies to items that are being pilot tested, even though they are not yet useful for scoring purposes.

Real cost. Another drawback associated with pilot testing is the real cost of piloting an item. Rudner (2010) places the overall development cost of a high-quality item at between $1,500 and $2,000. The cost of adding an item to an operational test form to understand its statistical properties may not be large relative to the final cost of the item. However, when considering the cost of pilot testing every item in an item bank over the entire lifespan of the assessment program, the costs are significant. Cumulative costs can be especially problematic when an item bank is maintained as part of a larger testing framework which utilizes CAT methodologies. If researchers could predict item

characteristics before their administration, testing organizations could avoid the costs of item piloting.

Item Difficulty Modeling

For an item to be used on a test form, evidence must be compiled that supports the adherence of the item to the construct of interest (American Educational Research Association, American Psychological Association, & Joint Committee on Standards for Educational and Psychological Testing, 2014). After an item is deemed appropriate for use by subject-matter experts (SMEs), an item is ready to be pilot tested. Under an IRT modeling framework, a high-quality item's difficulty parameter plays a large role in its selection for use on an assessment. Because of the weight put on item difficulty, researchers attempting to predict item statistical characteristics have more heavily focused on item difficulty than any other item statistic. The process of predicting item difficulty is called item difficulty modeling (IDM). Successful IDM would allow testing organizations to reduce the costs outlined above and realize several benefits.

If IRT item difficulty parameters can be accurately predicted for the target population of an assessment instrument, high-quality items can be used to construct assessments with less pilot testing. To make pilot testing unnecessary, item difficulty parameter predictions must exceed some accuracy threshold. Bejar (1983) places the threshold after which piloting becomes unnecessary at a .80 correlation between empirical and predicted item parameters. In practice, the threshold that would make item piloting unnecessary likely depends on the stakes of the specific testing situation. While the item difficulty parameter prediction accuracy threshold may be specific to the context

of the operational testing situation, exceeding the threshold could allow test developers to release test forms comprised nearly entirely of newly developed items while remaining confident of the test form's overall difficulty relative to the testing population.

There may also be benefits of IDM at sub-threshold accuracy levels. Predictions of moderate to high accuracy could be used as starting values when estimating IRT item difficulty parameters. Reasonably accurate difficulty parameter predictions could be used as priors in a Bayesian estimation framework. Using priors with Bayesian estimation methods has been shown to improve estimation, especially in small sample situations (Ames & Smith, 2018). At low accuracy levels IDM predictions could provide extra information to form builders when they construct pilot test forms. In some cases, piloting is not an option at all (ongoing assessment, small target population, etc.). In these cases, even modest predictions could prove useful.

Item writers can use predictions from IDM systems to improve the difficulty targeting of early item drafts that could decrease item editing and thus increase item retention rates. Moving beyond human item drafts, IDM systems could be incorporated into AIG systems to predict the difficulty parameters of automatically generated items that could not otherwise be feasibly pilot tested. Lastly, IDM techniques could be extended to other item statistics (e.g., item discrimination, or other item quality indicators).

Manual efforts. Initially, researchers wishing to realize the benefits of IDM attempted to leverage the knowledge of SMEs to predict item difficulty. SME item difficulty prediction studies produced mixed results. Lorge and Kruglov (1953) found

that when 2 groups of SMEs were not given any information about a set of 45 8th and 9th grade mathematics items, they were able to rank-order items by their relative difficulty, but not able to accurately predict the proportion of examinees who respond correctly (*p*-value). When given the *p*-values of 10 of the 45 items on a test, the SME groups could predict the average difficulty of the other 35 items within 2% and 12% of the true difficulty (individual SME item prediction errors were not reported). The result of the Lorge and Kruglov (1953) study is somewhat promising except for two caveats: 1) there was large variability across the estimates of the two groups of SMEs even when given *p*-value information, and 2) each SME group was comprised of seven SMEs. Needing large groups of judges to attain variable item difficulty predictions reduces the operational utility of predicting item difficulties in the first place.

Another study published thirty years later using the Test of Standard Written English (TSWE) found "that even after an extended period of practice and training, the accuracy in estimating item statistics of four subject matter experts does not approach the level that would be required to substitute ratings of item statistics for pretesting" (Bejar, 1983, p. 307). Bejar (1983) also found that SMEs couldn't explain exactly why the items on the TWSE varied in difficulty.

This early work suggests that SMEs are sub-optimal item difficulty predictors. So much so that in a 1998 work Impara and Plake raised questions about the results of standard setting procedures on the whole (Impara & Plake, 1998). Evidence for a measurable link between item features and item difficulty must be shown to exist if researchers exploring IDM can hope to model item difficulty from item features. To

identify links between item features and difficulty, researchers began exploring techniques alternative to SME expertise.

Motivated by a study finding that assessments could be equated using collateral item information (Mislevy, Sheehan, & Wingersky, 1993), Sheehan and Mislevy (1994) conducted a study finding binary classification trees (CART, Breiman, Friedman, Olshen, & Stone, 1984) could be used to partially explain the variation in the item difficulty parameters of the three parameter logistic IRT model. CART models recursively split the item feature space into regions, then fit simple models to each section of the feature space (Hastie, Tibshirani, & Friedman, 2009). CART models are covered more thoroughly in the predictive modeling section of Chapter 2. The study used SMEs to code the features of 114 mathematics items used in the Praxis I: Computer Based Test (CBT) pretest developed by the Educational Testing Service (ETS). The CART models were able to account for "36% of the variation in item difficulty parameters" (Sheehan & Mislevy, 1994, p. 12). Building a model that explains 36% of the variation in item difficulty statistics is a non-trivial achievement. However, using SMEs to code the items manually is resource intensive and 36% is not enough explained item difficulty variability to allow for a move away from item piloting.

When researchers manually create item features, they typically employ specific *a priori* hypotheses about which features of an item will affect its difficulty. The *a priori* approach forms the basis by which other forms of principled assessment design have been used to create groups of items with similar properties from a single item model, or template (Daniel & Embretson, 2010; Luecht, 2013; Mislevy et al., 2003). Although this

research has shown promise as a basis to produce item models for use in AIG systems (Enright, Morley, & Sheehan, 2002; Gorin & Embretson, 2013), like human item difficulty raters, manual item feature coding could be difficult to employ on a large scale.

In a testing situation with a high degree of item development, researchers pay the up-front cost of deciding what features to extract (e.g., time spent). Then, researchers pay the time cost of initially coding all items with the predefined features, and again each time a new item is introduced. Manually coding item features every time a new item is created is a reasonable approach for research purposes, but the added stress on the item development process would somewhat undermine its operational utility within large scale testing programs. Another potentially fruitful avenue in predicting the difficulty of items is automated IDM. Automated IDM is the process of building computerized systems to predict the difficulty of items without human input (other than initially building the system). Huff (2003) summarizes the advantage of automated IDM over manual feature coding well: "although such detailed variables [manually coded variables] do improve the modeling of item difficulty, the level of expert resources required to code items in this manner is extensive; thus, a promising area for future research is the development of software that can automate the variable creation process." The present research attempts to answer this call.

Lessons from automated essay evaluation. Automated essay evaluation is a line of research that can help put automated IDM studies into context. Automated essay evaluation engines are computer systems that score written standardized test essays (see Shermis & Burstein, 2013). Automated essay evaluation provides a good example of

what language related tasks can be automated with modern natural language processing (NLP) technologies.

Many automated essay evaluation engines train predictive models by extracting carefully selected linguistic features from essay responses (Shermis, Burstein, & Bursky, 2013). Automated essay evaluation engines use NLP software to convert polytomously scored (e.g., 1-5) essay responses into sets of predictor variables. Then, the essay evaluation engines generally use the NLP generated predictor variables as training data for predictive models. The trained predictive models are then used to score unscored essay responses. The training essays are collected by pilot testing specific essay prompts (Shermis & Hamner, 2013).

The general feature extraction, model training, and score (predict) approach that automated essay evaluation engines take will be mirrored in the present study. While procedurally similar, the present automated IDM study relies on different techniques than used to build and train an automated essay evaluation engine. In the present study the features extracted are not specifically chosen, the predictive model training data are piloted test items, and the predictions are continuous item difficulty parameters.

Overview of the Present Research

The present research compared predictive modeling methods on their ability to model the relationship between item difficulty statistics and automatically generated item difficulty predictor variables. The method used to automatically generate item difficulty predictor variables is novel to the IDM literature. The operational setting for the present study is a certification program in a scientific field. Specifically, the item bank used in

this study was comprised of items used on the American Board of Pediatrics (ABP) General Pediatrics certification (GP) and maintenance of certification (MOC-GP) examinations from 1999 to 2017 (see Appendix A for an example item, American Board of Pediatrics, 2018). The method for automatically generating predictor variables from test items outlined in this paper is meant to be applicable to many assessment situations using multiple choice (MC) items. However, the findings of this study may not generalize to other operational contexts.

The following chapter is a literature review that first discusses then presents relevant automated IDM literature. The literature review then introduces the distributed language representation framework and explains how it can be used to generate item difficulty predictor variables within an automated IDM context. Finally, the literature review describes predictive modeling study design considerations, the predictive modeling methods used in the study, and introduces the present study's research questions. After the literature review, Chapter 3 outlines the methodology used to address the study's research questions. Chapter 4 presents the results of the study. The final chapter discusses the results of the study, their implications, the limitations of the study design, and suggests future IDM research.

CHAPTER II

LITERATURE REVIEW

The following literature review has three main sections. The first section presents research that has explored automating the feature coding process for IDM. The most recent iteration of automated IDM research has come to exist at the intersection of psychometrics and computational linguistics/NLP. The first section of the literature review explains the roots of automated IDM within the field of psychometrics and how those roots have motivated researchers to draw on techniques and technologies developed by computational linguists and statistical learning methodologists.

The second section describes an NLP method for creating item variables from an item's text. The NLP method introduced in the second section is novel to the IDM literature. The third section presents predictive modeling methods for the present study. The research questions are introduced at the end of the chapter.

Automating the Feature Coding Process

Early studies incorporating automated feature coding only partially relied on computer automation. When modeling the *p*-values of an online English as a second language (ESL) test's reading and listening comprehension items, Rupp, Garcia, & Jamieson (2001) automatically extracted a subset of item features with computer software. The item features were total word count, average sentence length, the ratio of unique words to total words, and information density. Information density was defined as

the “number of phrases that consisted of a noun plus an attributive adjective plus a prepositional phrase” (Rupp et al., 2001). The regression model fitted to the Rupp et al. (2001) data was able to explain 31% of the test item difficulty variability. Each of the item features extracted in the Rupp et al. (2001) study is a logical driver of item difficulty given the ESL testing context. However, research hasn’t shown that linguistic complexity features are responsible for difficulty outside of the language testing context. Linguistic complexity should not influence the scores of doctors taking a GP certification examination.

Later studies began to incorporate item distractor choices, incorrect options, into the automated IDM equation by adding a measure of similarity between the distractors and the key. Hoshino & Nakagawa (2010) incorporated item distractors when modeling the item difficulty of fill-in-the-blank items on an ESL test. Hoshino & Nakagawa (2010) split a set of items into two groups (“easy” and “hard”) and correctly labeled 70% of the items in a two-way classification task. However, simple key/distractor similarity may not be expected to be as predictive of item difficulty on GP certification examinations.

Still in the language testing field, other researchers have applied automated techniques for feature extraction to generate large numbers of item features. Loukina, Yoon, Sakano, Wei, and Sheehan (2016) extracted features from 9,834 listening items used on an English language proficiency test. The extracted feature set was 339 features strong and primarily comprised of features related to text complexity; cohesion, discourse, syntax, and vocabulary. Loukina et al. (2016) extracted their 339 features using an automated NLP system called TextEvaluator (Sheehan, Flor, & Napolitano,

2013; Sheehan, Kostin, Napolitano, & Flor, 2014). Despite being extracted from listening items on an English language proficiency test, the text complexity features are general and meant to be usable "across a wide range of items" (Loukina et al., 2016). The generalized approach to thinking about IDM is a good example of what separates the automated methods from the more traditional methods described in the introduction where researchers and SMEs are required to code features in a context specific way.

Loukina et al. (2016) modeled item difficulty and SME item difficulty rankings with the extracted test complexity feature set using ordinary linear least squares regression, the regularization methods least absolute shrinkage and selection operator (LASSO) regression and elastic net regression, CARTs, *k*-nearest neighbors regression, stochastic gradient descent regression, linear and non-linear support vector regressions, and random forest regression. The regularization methods are designed to penalize complex models to reduce overfitting. Stochastic gradient descent regression uses the gradient decent optimization algorithm instead of the ordinary least squares approach. The gradient decent optimization algorithm is explained in the predictive modeling section of this chapter. The support vector regressions attempt to create functions that fit the training data with a pre-specified tolerance for error. Random-forest regression uses aggregated versions of many CART models, some with randomly initialized splits, to make predictions.

Because many of the predictive methods used in Loukina et al. (2016) are data mining methods that are prone to overfitting data, the item data set was randomly split into three data partitions (50% for training, 25% for model validation, and 25% for final

testing). Overfitting a data set is generating predictions that are not generalizable to new data from the same data source. A training data set is used to create initial predictive models. A validation data set is used to assess the fit of the trained model to new data. The validation set allows researchers to modify the initial model before finally assessing the model on the test data set. By training on a portion of the data and assessing overfitting with the validation data set, researchers can be more confident that the predictions of their final models on the test data set reflect their ability to generalize to new data. An explanation of relevant data partitioning methodology and its relationship to controlling overfitting is described in the data partitioning section of this chapter.

Random forest regression was the best performing method in the Loukina et al. (2016) study achieving correlations with SME rankings as high as .50 and with empirical item difficulty as high as .44 on the test set (25% and 19% variability explained respectively). While 19% is reasonable in an English language proficiency context, it would be a very high level of explained item difficulty variability for a text complexity feature set in GP certification setting.

Recently, Becker and Masters (2017) used a primarily automated approach to extract 20 features from 42 passage-based multiple-choice questions used on a university admissions examination taken in the United Kingdom. The study used R (R Core Team, 2017) to extract features automatically. The automatically extracted features were related to overlap between the passage and items, readability, word count, concreteness, word familiarity, and negation. Becker and Masters (2017) also manually tagged item type and

included SME ratings as additional features of their items. Unfortunately, the study could not explain any of the variability in item difficulty.

The present research is modeled most closely after a different study that used data mining methods to model item difficulty from automatically extracted item features. McLeod, Butterbaugh, Masters, and Schaper (2015) chose and extracted features from a set of standardized test items and accounted for 38% of the total variability of item difficulty in the training set using a CART model. Their study did not account for a significant proportion of variance in the test set. However, they did not partition the item features into training, validation, and test sets, instead using only training and test sets. Leaving out a validation set may explain the issues the study encountered when accounting for item difficulty variability in the test set as it is difficult to detect overfitting without a validation set. A close review of the study is conducted here.

In their research study McLeod, Butterbaugh, Masters, and Schaper (2015) extracted 177 features from 448 MC licensure examination items. The features extracted were either related to sentence structure, readability, parts of speech, or verb tense. The authors treated the item stems and keys separately but aggregated the distractors before extracting 59 features from each of the three parts of each item to arrive at the 177 total features. The features were extracted using the Python Natural Language Toolkit (NLTK; Bird, Klein, & Loper, 2009) an open source library of tools for NLP written for use in the open source programming language Python (Python Software Foundation, 2017). The authors split the items into a 300-item subset for model training, and a 148-item subset for model testing. McLeod et al. (2015) used CART and multivariate adaptive regression

splines (MARS, Friedman, 1991) to model item data. MARS is a method akin to a generalized form of forward stepwise linear regression (Hastie et al., 2009). MARS creates piecewise functions for each input variable then uses linear combinations of the piecewise functions to model a target variable.

In the model training phase of their study McLeod et al. (2015) were able to train a CART model that explained 38% of the total variability in the Rasch item difficulties of the training item set. The Rasch model (closely related to IRT models) is a single parameter model that estimates and places item difficulty and person ability parameters on the same latent trait scale (Rasch, 1961). Interestingly, The MARS model was only able to explain 20% of the Rasch item difficulties in the training item set. Since the MARS model could not explain as much variance as the CART model in the training set, and due to "the complexity of the multivariate spline model," the study abandoned the MARS method altogether (McLeod et al., 2015). In any case, once applied to the test item set, the CART model was unable to explain any variance in the Rasch item difficulties. The CART model's poor accuracy illustrates the effects of a model which has been overfit to training data. Explaining far less, or no, variation in test set relative to the training set is exactly what one would expect of an overfitted model.

While this study provides a good example of the typical process of automated IDM, the authors point out a few limitations which undermine the study and potentially explain the CART model's inability to account for variance in the test item set. First, as the Rasch model is meant to account for the covariance in item performance and attribute that covariance to the underlying latent trait, the authors may have only capitalized on

chance in the training set (McLeod et al., 2015). This explanation makes sense given the author's focus on item syntactic structure when extracting features. The only non-syntactic feature of the items used in model building was an item content code. The items used in the study came from an insurance certification test, not a language test. In anything other than a language testing context, researchers may be justified in expecting syntactic features to be poor drivers of item difficulty.

Next, the authors explain that they may have simply overfit the CART model to the training set (McLeod et al., 2015), a conclusion which seems especially likely because the initial item set was only split into two subsets. Splitting the study's data into only two subsets makes it impossible to know which modeling decisions made during training resulted in model overfit when making predictions with the model on the test set.

Lastly, the authors inferred that they may have needed larger samples "to adequately fit the complex feature space" when using the CART method. The complexity of their feature space was reduced by aggregating distractors. However, aggregating the distractors may not have produced enough of a reduction in complexity given the sample size. The lessons from the automated IDM studies reviewed serve as the drivers for the present study's methodological design.

Distributed Representations of Language

The research literature suggests that an item's difficulty is a function of the interaction between the ability of the population taking it, error, and its components (stem, key, distractors, etc.) (Hoshino & Nakagawa, 2010; Loukina et al., 2016; McLeod et al., 2015; Rupp et al., 2001). To model item feature interactions, an IDM study must

first convert the components of an item into a numeric vector of predictor variables which represent the items. After creating a representative vector, it can be used as the training data for predictive models. The first task of the IDM researcher wishing to automate IDM is to automatically encode representative vectors for the item components. This section 1) introduces distributed representations, a theoretical framework for encoding item components whose application is novel to the automated IDM literature; 2) describes a family of methods for obtaining distributed representation vectors; and 3) argues why distributed representations of items may be useful in automated IDM applications.

One method for automatically encoding representations of items is by using a localist representation. A localist representations is one in which "one computing element [is used to represent] each entity" (Hinton, McClelland, & Rumelhart, 1986, p. 77). An example of a localist representation is a one-hot encoding for a word with a vector that has an element for each word in the context's vocabulary. In one-hot encoding the element corresponding to the word of interest is coded "1" and all other words in the vocabulary are coded "0." One-hot encodings can result in especially sparse input data for tasks that deal with language data. For example, even extending one-hot encodings to the sentence level or higher—coding word counts for each of the words in a sentence or paragraph—would result in too sparse a representation space for the predictive modeling frameworks described in the predictive modeling section below. Applying one-hot encoding to an entire item bank for an automated IDM task would result in orders of magnitude more variables than items for training predictive models. Also, most variables for any given item would be coded "0" providing only sparsely distributed information

patterns for predictive models to characterize. A more compact method of representing language is through distributed representations.

The distributed representations of language theory is a framework for computer language understanding (Hinton et al., 1986). Under the distributed representation framework, text data is converted into vector representations of lower dimension than localist representations. The distributed representation framework hypothesized artificial neural network (ANN, Rosenblatt, 1958) architecture as a method to capturing the meaning of natural language in relatively low dimensional vectors. Basic ANN architecture is covered in the predictive modeling section of this literature review. The knowledge that ANNs are created by generating multiple layers of linear combinations is necessary to understand this section. In their simplest form, ANNs contain three layers connected through linear combinations and transformations. The first layer is called an input layer and is comprised of observed data records. The second layer is called the hidden layer and contains nodes. The number of nodes that make up the hidden layer is a hyperparameter of ANNs. Hyperparameters are parameters of models which can be adjusted before fitting the model to training data. Each node of the hidden layer linearly combines and transforms the input layer. The hidden layer is again linearly combined and transformed to create an output layer of predictions. ANNs can be used for either classification or regression tasks. Under the distributed representations of language framework, the hidden layer is used as a low dimensional representation of language units relative to one-hot encoding.

Representing language with low dimensional vectors is "efficient whenever there are underlying regularities which can be captured by interactions among microfeatures" (Hinton et al., 1986). Put another way, distributed representations operate under the theory that words that co-occur in text have similar meanings (Harris, 1954), and that co-occurrence patterns can be approximated with ANNs. Lower dimensional distributed representation vectors of language more naturally lend themselves for use in downstream tasks such as automated IDM by asking less of predictive models than one-hot encodings do.

Word2Vec. The Word2Vec models were a major advancement in the development of language models based on the distributed representations framework (Mikolov, Chen, Corrado, & Dean, 2013; Mikolov, Sutskever, Chen, Corrado, & Dean, 2013)[1]. The Word2Vec models take advantage of basic ANN architecture to generate relatively compact distributed representations for words by training ANNs on large data sets.

The input and output data for the Word2Vec models are one-hot encoded written natural language (articles, books, etc.) of equal dimension to the size of the entire vocabulary. Like the input, the true output is also a one-hot encoded vector equal in dimension to the size of the vocabulary. The Word2Vec models are trained by moving across the training text data and using the words within a context window as the input

[1] The literature on distributed representations often calls them word, sentence, paragraph, or document *embeddings* or *vectors* (e.g., *word embeddings, sentence vectors*, etc.).

data. The context window is comprised of some number of words before and after the target word in a sentence. The number of words on either side of the target word that make up the context window is a hyperparameter of the model. The context window moves across the written language training data one word at a time until the ANN has been trained on all the training data.

Each of the two models that make up Word2Vec takes the opposite approach of the other in how an ANN model is trained. The continuous bag-of-words model (CBOW) trains an ANN classifier to predict a target word given the words in the context window as inputs. The other Word2Vec model, the continuous skip-gram model (SG), trains an ANN classifier to predict words in a context window given a target word. Both CBOW and SG use a variant of the softmax function to transform the outputs. However, the distributed representations of the words in the vocabulary are not the outputs of the trained classification models. Instead, the vectors of derived features which make up the hidden layer of the trained ANNs for each word are used as the distributed representations for the words in the vocabulary.

Representation meaning and quality. After training ANNs on a corpus containing around 6 billion words, the distributed representations generated from the CBOW and SG models had useful properties. For instance, words that are conceptually related were grouped together in multivariate space. The names of countries like "France" and "Italy," and capitals like "Paris" and "Rome" grouped together. While multivariate groupings serve as evidence that the Word2Vec models encode at least similarity, the authors found that when algebraic operations were performed on the

representation vectors the resultant vectors were also meaningful. The manipulability of the Word2Vec models is likely due to the finding that the models implicitly factor the matrix of original input data (Levy & Goldberg, 2014).

For example, Word2Vec completed syntactic and semantic analogies tasks by employing simple vector arithmetic. The authors designed 10,675 syntactic and 8,869 semantic analogies tasks by pairing sets of manually created analogies based on category groupings. Syntactic analogies had relational categories like adjective to adverb (apparent/apparently paired with rapid/rapidly) and comparative (great/greater paired with tough/tougher). Semantic analogies had relational categories like common capital city (Athens/Greece paired with Oslo/Norway) and currency (Angola/kwanza paired with Iran/rial).

To complete the comparative syntactic analogy task with pairs big/biggest and small/smallest without the word "small" the Word2Vec vector for the word "biggest" was subtracted from the vector for the word "big" and added to the vector for the word "smallest." The result of the simple vector arithmetic calculation produced a vector which is closest by cosine distance (cosine similarity) to the vector for the word "small." Cosine similarity is a vector similarity measure calculated by dividing the dot product of two vectors by the product of their magnitudes.

$$cosine\ similarity = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} \quad (1)$$

Cosine similarity ranges from -1 to 1. A mean-centered cosine similarity produces a Pearson correlation. When searching based on greatest cosine similarity, the Word2Vec CBOW and SG models correctly recovered the exact word that completed syntactic analogies 64% and 59% of the time respectively. The SG model also performed well on semantic analogies tasks, such as the relationship of "France" to "Paris" and "Germany" to "Berlin" as well as "king" to "man" and "queen" to "woman," accurately recovering the exact word 55% of the time. The CBOW model only recovered the correct word 24% of the time on the semantic analogies task.

The SG model was also tested on a 5-choice MC sentence completion task. On the sentence completion task the SG model accurately chose the correct answer choice 48% of the time. Performance on both types of analogies tasks and the sentence completion task suggests that the SG model encodes word meaning in its vector representations at a level which may be of high enough quality to be useful for extracting features from MC item components in an IDM context. However, an extension must be made before the Word2Vec SG model can hope to be useful for automated IDM on the GP certification examination questions or other questions in scientific fields: handling out-of-vocabulary (OOV) words. OOV words are words that a model has not been exposed to during training. The GP certification examination has many rare words that are unlikely to exist in any but the most context specific training language data sets.

FastText. Despite the Word2Vec SG model's potential utility for automated IDM, it has no way of handling OOV words. Rare terms-of-industry may drive item statistics in applications of automated IDM to scientific fields such as the present study.

If rare words in technical fields do influence item statistics, and are OOV, then variance will be unaccounted for when generating representations for item components even before predictive modeling takes place. FastText is a pair of models that generalize the Word2Vec models by incorporating subword information (Bojanowski, Grave, Joulin, & Mikolov, 2017; Mikolov, Grave, Bojanowski, Puhrsch, & Joulin, 2018). Incorporating subword information when generating representations allows FastText to naturally handle OOV words.

FastText creates distributed representations at the subword level using the same framework as the SG and CBOW Word2Vec models with the addition of creating representations for character *n*-grams. Character *n*-grams are combinations of *n* adjacent letters in a word. The context window is moved across whole words as well as across character *n*-grams when training the FastText models. Moving the context window across character *n*-grams enables the FastText models to automatically pick up morphemes. Morphemes are the smallest units of a language that still carry meaning. Many of the Greek and Latin roots that make up words in English are morphemes of English. To arrive at the representations for a word, FastText averages the representations for the *n*-grams in the word with the representation for the word itself. Extending FastText's property of combining known *n*-gram vector representations to OOV words allows researchers to generate representations for any word regardless of whether it was seen during training. FastText SG will be the focus of the present study.

The SG architecture is more suited than the CBOW architecture to handle the rare words found in the present study's scientific examination context. For example, if a word

is rarely used in a phrase, in training SG will still be able to predict the context. In contrast, if a word is rarely used in a phrase, in training CBOW will assign a very low probability to the word when that phrase comes up.

Representation meaning and quality. The improvements of FastText over Word2Vec cannot hope to be useful for automated IDM unless FastText is shown to generate meaningful word representations. The original study evaluated FastText SG with a series of experiments to provide evidence that it produces meaningful word representations (Bojanowski et al., 2017). A few of the experiments done with FastText SG are relevant to the present study.

One experiment compared the relationship between FastText SG word representation similarities to human ratings of word similarities (Bojanowski et al., 2017). The cosine similarities of word representations generated by FastText SG correlated more highly with human word similarity ratings than Word2Vec SG and CBOW in 9 out of 10 word similarity data sets across 7 languages (three German, two English, a French, a Spanish, an Arabic, a Romanian, and a Russian dataset). FastText SG also correlated more highly with human word similarity ratings than other representation methods that also account for morphemes when creating word representations (Botha & Blunsom, 2014; Luong, Socher, & Manning, 2013; Qiu, Cui, Bian, Gao, & Liu, 2014; Soricut & Och, 2015).

Like Word2Vec, FastText SG was tested on analogy completion (Bojanowski et al., 2017). On semantic word analogy tasks, FastText SG performed about as well as the Word2Vec models. On syntactic word analogies tasks, FastText SG performed better

than Word2Vec, correctly completing English analogies 75% of the time compared to 70% of the time for each of the Word2Vec models. The results of the experiments on model-human similarity ratings and analogy completion speak to the quality of word representations generated by the FastText SG model with respect to word similarity. Another study found positive results in a setting which more closely matches that of the present study.

Scientific language in medicine. Ezeiza Alvarez (2017) compared word representations in a medical language context. Specifically, Ezeiza Alvarez (2017) trained and tested representation models on the Unified Medical Language System (UMLS) Metathesaurus, "a repository of inter-related biomedical concepts" (Bodenreider, 2004, p. D267). Ezeiza Alvarez (2017) used a triplet task to evaluate FastText's representation quality.

The triplet task started by creating 3,649 related word pairs from the UMLS Metathesaurus and adding a third word to each pair chosen at random. Then, the similarity of FastText representations between the two related words was compared to that of one of the related words and the randomly chosen third word. Accurate performance on the triplet task was calculated as the frequency with which related words were found to be more like one another than either of them was to the randomly chosen word. The idea behind the triplet task is that a representation model which creates meaningful word representations will generate more similar representations between known to be related words than between two randomly selected words.

When training on 1B words from the UMLS Metathesaurus, FastText SG performed more accurately than the Word2Vec models on the triplet task (93% versus 83% for Word2Vec SG and 85% for Word2Vec CBOW). Greater separation between FastText SG and the other models was observed when training to create representations for 1M words (81% versus 67% for each Word2Vec SG and CBOW) suggesting better utility in smaller samples. However, there is a caveat to the findings. The results of the Word2Vec models on the triplet task when trained on 1M words apply to known words only.

Because the Word2Vec models have no way of handling OOV words, when trained on 1M words they could only be evaluated on 5% of the triplets as the other 95% contained at least one OOV word. When considering the entire triplet data set, the Word2Vec SG and CBOW models' accuracy drop to 4% and 3% respectively. What is striking about considering OOV words is that for the results of the 1M word training set, FastText SG had to rely on subword information to create representations for at least one word in 95% of the triplets yet managed to achieve 81% accuracy.

The above findings are more impressive still when considering that they were all found using a stemmed training corpus. Stemming words, a form of language normalization, is a practice used to decrease the variability of language data. Stemming removes suffixes from root words before training language models (e.g., "es" is removed from "foxes" to keep only the root word "fox"). One benefit of stemming before training distributed representation models is that stemming results in a single representation for a word instead of a separate representation for each conjugation of a word. FastText

naturally needs less normalization because it accounts for subword information. Accounting for subword information makes stemming unnecessary and possibly detrimental to the quality of the final word representations when using FastText. Ezeiza Alvarez (2017) infers that "it is very likely that FastText would [have] compared much better if we [*sic*] were working with non-normalized words" (p. 43).

Ezeiza Alvarez (2017) attempts to explain the success of FastText SG on the triplet task by positing that "it is possible that FastText may be specially effective on scientific text, as most technical words are derivative and share many of the same subword components" (p. 43). Like scientific text generally, test items which tap knowledge, skills, and abilities (KSAs) in medical contexts are likely to have many rare words which are structured logically and often comprised of Greek and Latin morphemes to form organized complex (multi-morphic) words. Positive results using FastText SG on medical literature provided an indication that FastText SG could be leveraged in the present study to create useful representations for item components containing medical language and concepts. Nonetheless, a discussion of some of the drawbacks of FastText SG is warranted.

Potential issues with FastText SG. Using FastText SG for automated IDM should be done with a few considerations. As with all ANNs, the size of the hidden layer (number of nodes) is a hyperparameter of the distributed representation models. Because the hidden layer of the ANN becomes the distributed representations, the hidden layer of the generating ANN and the resultant distributed representations are of equal dimension. In the original studies introducing Word2Vec and FastText, 300-dimensional vectors are

the size at which the authors found diminishing returns in the model testing tasks (Bojanowski et al., 2017; Le & Mikolov, 2014; Mikolov, Chen, et al., 2013; Mikolov et al., 2018). However, 300-dimensional word representations may be difficult to handle when used in an automated IDM study. The results of experiments with FastText SG on medical language were achieved with models trained to create 100 dimensional representations (Ezeiza Alvarez, 2017). While an exploration of the effects of dimension size on item component representation quality would be interesting, it is beyond the scope of the present study.

Another obvious issue with the Word2Vec and FastText models is that they completely ignore word ordering (hence the name bag-of-words, BOW). High-stakes test questions are written to very exact specifications and it is hard to imagine that the order of the words in the components of a test question do not influence the question's statistical characteristics. While more sophisticated language modeling techniques for modeling sentence structure exist, they are also beyond the scope of the present study. Higher-order language models may not be necessary if basic distributed word representations can do the job of representing item text in an IDM context.

Lastly, the Word2Vec and FastText models do not address the question of exactly how a researcher is to combine word representations for item components with more than one word. In the present study's context, many of the item components—at least every item stem—are comprised of multiple words. Also, some words in an item's components are likely to contribute more to item difficulty variability than others. A researcher combining word vectors haphazardly would be smoothing over potentially rich sources of

item difficulty variability from individual words in the item components. Extending word representations to sentences, paragraphs, and larger units of language while measuring and accounting for word importance is addressed in the following section.

Higher order language units. Evidence for the quality of combined vector representations produced by the FastText SG model is necessary to warrant serious consideration for use in IDM. The desire to extend word representations to higher order language units to mitigate the obvious issues with BOW models motivated research which extended Word2Vec almost immediately after its development (Le & Mikolov, 2014). To create higher order representations, identifiers were added to Word2Vec during training to generate representations for paragraphs and documents. When used as inputs to a logistic regression model, Le & Mikolov (2014) found that sentence representations could classify sentiment (positive/negative) with 12.2% error, less than any of the other 7 language representation models compared. Similarly positive results for classification tasks were observed when simply averaging Word2Vec (White, Togneri, Liu, & Bennamoun, 2015) and FastText SG word representations (Joulin, Grave, Bojanowski, & Mikolov, 2017). However, in an IDM context a researcher might expect to lose information when an item stem comprised of many words is averaged because less important words will dominate the final aggregated representation.

A method of accounting for word importance when averaging word representations is by adjusting them with term frequency-inverse document frequency weights (TF-IDF, Jones, 1972, 1973). For the present study, TF-IDF is defined as

$$tf\ idf(w) = tf * log\frac{N}{df(w)} \quad (2)$$

where tf is the number of times a word appears in an item, N is the number of items in the item set, and $df(w)$ is the number of items containing the word. In distributed representations of language research, TF-IDF weighting is supported by findings that rare words often best capture the meaning of a piece of text (Leskovec, Rajaraman, & Ullman, 2014) and has been shown to improve combined word representations (Arora, Liang, & Ma, 2017). Applying TF-IDF weights to test items increases the weighting of words within an item which are rare relative to the item bank. Although researchers have pointed out that "it is unclear to what extent the sentence's position in the vector space reflects its semantic meaning, rather than other factors such as syntactic structure" (White et al., 2015, p. 1), the present study attempted to offset some of the drawbacks of sentence representations by using TF-IDF weighted averages for item components.

TF-IDF weights can be applied in IDM by first calculating them for each word in the item bank. Then, multiplying each word vector by its corresponding TF-IDF weight resulting in weighted word vectors. Finally, the TF-IDF weighted word vectors that make up each item component can be averaged to create a single representation for each item component within an item.

The demonstrations of aggregating representations to model higher order units of language are used to complete relatively simple tasks. There is generally less evidence in more complicated semantic settings. There have also been recent arguments against the ability of distributed word representation to represent meaning (Camacho-Collados & Pilehvar, 2018). Because word representations are often trained on large natural language corpora, the different uses—or senses—of words are represented together in a single vector. As in the present study, aggregating many word representations may exacerbate the issue of compressing many word senses into single vector. The line of research attempting to disentangle the different senses of a word that are captured in a single representation vector is its infancy (Conneau, Lample, & Baroni, 2018). By aggregating word representation vectors the present research likely pushes the distributed representations of language framework to the edge of its present capabilities.

Predictive Modeling Methods

This section presents two sets of methodologies that were used together in the present study for training models to predict item difficulty parameters. First, a model cross-validation design is presented and rationalized. Then, a series of predictive methods are introduced.

Data partitioning and hyperparameter tuning. Many of the predictive modeling methods used in the present study are prone to overfitting data. To control overfitting, researchers adjust model hyperparameters. The process of adjusting model hyperparameters is called tuning.

Cross-validation designs are useful methods that allow researchers to assess overfitting and tune model hyperparameters. Model cross-validation methods allow researchers to understanding how well predictive models may be expected to generalize to new data.

The present study employs a variant of the hold-out cross-validation design procedure used in the data mining literature to reduce overfitting risk by creating a framework to appropriately tune model hyperparameters (Devroye & Wagner, 1979). In preparation for predictive model hyperparameter tuning, the data set of interest is split into three randomly selected subsets: a training set, a validation set, and a test set. Ripley (1996) gives the following definitions of training, validation, and test sets:

- **Training set** a set of examples used only for learning, that is to fit the parameters of the classifier.
- **Validation set** a set of examples used to tune the parameters of a classifier…
- **Test set** a set of examples used only to assess the performance of a fully-specified classifier. (pg. 354)

As its name suggests, the training set is used to train each predictive model. Likewise, the use of the test set is to test each model's predictive accuracy on data that has never been seen by the model. The validation set plays a subtler role in the process of building predictive models.

Like the test set, the validation set is never used to train a predictive model. The validation set is an intermediary data set that is used during model building to reduce the potential of overfitting when tuning each predictive model's hyperparameters. The validation set is valuable because when tuning each predictive model with only the training data, a researcher has no way of knowing whether improvements in predictive accuracy are due to the correspondence of the model to the true relationship between the predictor and target variables or due to capitalization on chance. Continuously choosing the version of a model whose predictions are most predictive of the training data may lead a researcher to choose a model that is overly dependent on the unique features, or error variance, of the training set. The validation set helps solve the problem of how to tune model hyperparameters.

To summarize the hyperparameter tuning process, the training set is used to initialize a model. Predictions are then made on the validation set to identify possible issues with overfitting. Next, adjustments to model hyperparameters are made until the model performs as accurately on the validation set data as possible. Lastly, predictions are made on the test set using the final version of a model (Ripley, 1996).

Using strict model cross-validation designs is essential to understanding data when working with algorithmic modeling techniques. The nature of many data mining methods leaves them susceptible to overfitting data. Studies which do not properly validate their fitted models undermine their findings. Model cross-validation designs also open the door for comparing more aggressively fitting predictive methods by creating an environment in which model overfitting can be more easily characterized and reduced.

Predictive modeling. Once the data set is partitioned, predictive models must be chosen to fit to the training and validation data sets. This section provides descriptions of the predictive methods used in the present research. Each predictive method has its own set of features that make it uniquely promising for IDM. Throughout the section, the reader should keep in mind that population inferences take a back seat to predictive accuracy in the present automated IDM study. For ease of interpretation, this section uses the statistical notation presented by Hastie, Tibshirani, and Friedman (2009) across all predictive modeling methods.

Linear least squares regression. Linear least squares (LLS) regression—often called ordinary least squares (OLS) regression—is a frequently used approach to predictive modeling. The basic features of LLS regression are introduced first in this section. Then, LLS is discussed within the context of an automated IDM study.

In its simplest form, LLS regression assumes a linear relationship between the output variables Y and the observations i of the p input variables $X^T = (X_1, X_2, \dots, X_p)$. This linear relationship assumes the form

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{3}$$

such that $f(X)$ is the predicted value of Y from the sum of the intercept β_0 and the sum of the products of the observed value of the input variables, X_j, and their regression coefficients β_j. LLS regression coefficients are typically estimated by the method of least squares.

Least squares estimation minimizes the residual sum of squared deviations (RSS) of output y_i and input x_{ij} across N total observations

$$RSS(\beta) = \sum_{i=1}^{N}(y_i - f(x_i))^2 = \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \quad (4)$$

where β "is a vector in input space that points to the steepest uphill direction" (Hastie et al., 2009, p. 12) comprising an estimated coefficient for each X_j.

LLS regression's interpretability makes it useful to include in studies attempting to model item difficulty. When the data are standardized, the value of β_j corresponds directly to the amount of unique variability explained by X_j. The correspondence to variance explained is conducive to a straightforward understanding of variable importance when multiple input variables are included in an analysis. Some of the other predictive methods described below leave much to be desired in terms of interpretability by trading it for potential gains in predictive accuracy.

Also, when no inferences about the population are being made—as is the case for the present automated IDM study in which generalizable predictive accuracy is the primary objective—LLS regression makes no firm assumptions about the data beyond the existence of linear relationships. LLS regression simply calculates a line of best fit to the data. The trade-off is that despite being easy to implement, interpret, and use, linear models (LLS regression or otherwise) would be unlikely to provide the best predictions when modeling a large and complex input space of item linguistic features. The linearity property of LLS regression is especially troublesome when there are highly correlated

input variables in X^T. The next two predictive methods, principal components (PC) regression and partial least squares (PLS) regression, attempt to mitigate the problems generated by highly correlated input variables by utilizing what Hastie et al. (2009) call "derived input directions" (p. 79). These methods use the original p inputs X_j to create linear combinations $Z_m, m = 1, \dots, M$ of X_j where m is the number of linear combinations. The derived input methods get their name because they then use Z_m as inputs for regression models instead of X_j.

Principal components regression. PC regression handles large numbers of input variables by aiming to create a composite summary of X_j. PC regression achieves this by reducing the dimensionality of the X_j by creating orthogonal (i.e., uncorrelated) linear combinations $\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_m$, called principal components, of the matrix of input variables X_j such that each linear combination is orthogonal to the next. Principal components that maximize the variability in X_j are successively created by multiplying each eigenvector v_m of the centered input variables matrix $\boldsymbol{X}$ by $\boldsymbol{X}$

$$\boldsymbol{z}_m = \boldsymbol{X}v_m. \tag{5}$$

That is, the first principal component $\boldsymbol{z}_1$ of $\boldsymbol{X}$ is calculated and is the projection, or direction, of $\boldsymbol{X}$ which has maximal sample variance. Next, conditional on being orthogonal to $\boldsymbol{z}_1$, $\boldsymbol{z}_2$ is calculated and captures the next most variance in $\boldsymbol{X}$. The algorithm continues until $\boldsymbol{z}_m$ is calculated, which captures the minimal amount of sample variance in $\boldsymbol{X}$ and is orthogonal to all other principal components.

After $\boldsymbol{z}_m$ is calculated, a subset of principal components is retained for entry into a regression equation as predictors starting with the principal component corresponding to the largest eigenvalue, λ, according to some cut off rule. One option is to retain all principal components that account for some value of variance in $\boldsymbol{X}$. However, retaining all principal components would be equivalent to using X_j in its entirety. Cattell (1966) offers another simple method for selecting the number of principal components to retain in subsequent analyses. Cattell (1966) developed a plot, called a scree plot, which plots λ_m against m. The researcher is advised to retain all principal components before the values of λ begin to flatten out. While this is a somewhat subjective rule of thumb, it can help researchers select a minimal number of principal components which do a good enough job of capturing the major sources of variability in $\boldsymbol{X}$. Another method is to continually add principal components until additional components degrade performance on the validation set. The second approach will be the one used in the present study.

Like the methods described below, and in contrast to LSS regression, the advantage of using principal components instead of specific variables as inputs when developing predictive models for modeling item difficulty is that the principal components offer a systematic way to handle large numbers of input variables relative to sample size. PC regression is also straight forward to implement within the context of predictive modeling. Because the principal components are simply transformations of input variables, they can be generated with the training data then easily computed for the validation data. However, PC regression does come with drawbacks. An introduction of PLS regression will provide context for a potentially serious drawback of PC regression.

Partial least squares regression. The second method that uses derived input directions as inputs into a regression model is PLS regression. The PLS approach to regression was initially developed by Herman Wold in the 1960s (Wold, 1966). Like PC regression, PLS regression seeks orthogonal linear combinations of the predictor variables that explain the variance in the target. However, PLS regression also seeks linear combinations that explain the correlations of the inputs with the target variable (Frank & Friedman, 1993; Stone & Brooks, 1990). The tradeoff that PLS regression makes between optimizing for high correlations between the inputs and the target variable and optimizing for variance explained in the inputs is achieved by algorithmically using the target in the creation of PLS directions. The target correlation/input variance tradeoff is the major feature of PLS regression that differentiates it from LLS and PC regressions.

To create PLS directions, first $\hat{\varphi}_{mj}$ is obtained by calculating the inner product of each input with the target

$$\hat{\varphi}_{mj} = \langle \boldsymbol{x}_j^{(m-1)}, \boldsymbol{y} \rangle = \sum_{i=1}^{N} x_{ji} y_i. \tag{6}$$

Next, $\hat{\varphi}_{mj}$ is used to calculate each derived input $\boldsymbol{z}_m$

$$\boldsymbol{z}_m = \sum_{j=1}^{p} \hat{\varphi}_{mj} \boldsymbol{x}_j^{(m-1)}. \tag{7}$$

Then, the regression input coefficients corresponding to $\boldsymbol{z}_m$ are calculated by regressing $\boldsymbol{y}$ onto $\boldsymbol{z}_m$. Finally, each $\boldsymbol{x}_j^{(m-1)}$ and $\boldsymbol{z}_m$ are orthogonalized before repeating the process to obtain the next PLS input direction.

Like LLS and PC regression, PLS regression only assumes that a linear relationship exists between the inputs and targets of the modeled data. Like PC regression, PLS regression uses the input variables as well as the target variable to fit the data. PLS regression's emphasis on finding derived input directions which maximizing correlations as well as variance in the target allow PLS regression to fit data more closely than LLS and PC regression in most situations. Specifically, the advantages of PLS regression over LLS regression can be realized in cases such as the present study where there are many, possibly collinear, predictors with a relatively small number of observations. When conducting a PC regression with many predictors, many principal component directions are possible. However, a PC regression that selects a subset of the principal components based on those that best explain the variation in the predictors will not necessarily be the PC regression most predictive of the target (Hadi & Ling, 1998; Jolliffe, 1982).

In the context of the present study, the primary drawback of using PLS regression is that it is more likely to overfit the training data when compared to LLS and PC regression models. Potential issues with overfit highlight the importance of data partitioning for model cross-validation. The final two predictive methods can also tend to overfit data if not appropriately validated. However, they each take unique approaches to fitting data.

Classification and regression trees. A regression tree is a continuous outcome variable predictive model that first algorithmically splits a variable space into regions, then fits simple regression models to each region. The process is named after decision trees. In multivariate space the tree's "nodes" are the space's split points, the "branches" are the splits, and the "leaves" are the resultant regions. The algorithmic process of splitting the variable space into regions is called binary recursive partitioning. Specifically, binary recursive partitioning iteratively splits the feature space into pairs of rectangular regions.

To predict an outcome Y given observations x_i for inputs j_p, a constant c_m is calculated for each region R_m of the input space. Across the entire input space, the function

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m) \tag{8}$$

models Y. From $f(x)$ we see that simple constants for each region are all that a basic CART uses to make its final prediction of y_i. That is, a single value of the output in a specific region is used as the prediction constant $\hat{c}_m$ for region R_m. The difficulty of creating a CART model lies in the process of splitting the feature space into regions. The split point used to create R_m, and the variable it applies to, must be chosen at each binary recursive partitioning iteration. To create each split, every possible value of every predictor variable is considered. The value of the split is selected based on the value which minimizes the sum of squared deviations from the mean of the outcome variable in

that region of feature space. However, directly finding the value of the minimum sum of squared deviations from the mean of the output variable across all input variables to use as the binary split point, s, for a region can be computationally challenging. A greedy algorithm can be used to avoid the computational cost of searching through all sum of squared deviations values across each input variable by finding local minima. For regression trees, the greedy algorithm searches for the best combinations of j and s for each R_m which solve the minimization

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right] \tag{9}$$

where R_1 and R_2 are the half regions created from splitting R_m. The estimated constants which serve as the estimates of y for each half-region, $\hat{c}_1$ and $\hat{c}_2$, are

$$\hat{c}_1 = ave\big(y_i|x_i \in R_1(j,s)\big) \text{ and } \hat{c}_2 = ave\big(y_i|x_i \in R_2(j,s)\big). \tag{10}$$

Searching for the average of y_i given x_i within each half region to serve as $\hat{c}_1$ and $\hat{c}_2$ for those regions is a computationally effective way of building a regression tree. The algorithm can be repeated until each data point belongs to its own specific region of the tree. However, repeating the algorithm until a region is created for every data point would result in a model which is perfectly fit to the training data and unlikely generalizable to new data sets. A tree with as many regions as data points—having N regions—would be as complex as the underlying data and improbable to accurately predict any new

observation. Therefore, the size of the regression tree must be controlled to achieve a good balance of underfit and overfit.

Researchers avoid the overfitting issue by tuning a regression tree's hyperparameters. The primary hyperparameter of a regression tree is its size. A tree's size is defined by the number of branches it has. The two main ways to control a regression tree's size are stopping rules and pruning. Stopping rules are predetermined criteria which halt the binary recursive partitioning algorithm after a certain point (e.g., after 10 splits). Pruning removes branches on an already grown tree. Stopping rules and pruning can aide the researcher in striking a balance between underfit and overfit by first building a large, potentially overly complex tree, then pruning back nodes depending on the model's fit.

There are a few advantages of regression trees over other modeling methods. For instance, regression trees make no assumptions about underlying structural characteristics of the data. Operating with no assumptions about the data allows regression trees to be fit to a data set with any relationship between the predictors and the outcome. In fact, binary recursive partitioning can adjust to different relationships across a single data set's outcome space. Regression trees have no problem modeling different relationships between the predictors and the outcome at different levels of the outcome variable. In a typical LLS regression model a constant weight is calculated and applied globally for each input variable.

Regression trees are also easily interpretable. While not a specific benefit of regression trees relative to LLS regression, because LLS regression models are also

easily interpretable, the ability of a regression tree to fit a complex input space and remain interpretable is a valuable feature. Even missing data can be used in such a way that its contribution to the model can be easy to understand. A regression tree can use a missing value for an input variable as a split point. The final tree will simply show that a missing value for that input was used to split the data in that specific branch.

However, regression trees do have shortcomings, especially with respect to generalizability. If a regression tree is used to predict the outcomes in a new data set whose feature space is different than that of the data set used to train the tree, prediction error can become quite large. Large increases in error when applying a regression tree to new data is introduced by the binary recursive partitioning algorithm used to build it. Each subsequent branch relies on the accuracy of the last, so errors can propagate through subsequent branches of a tree. Thoughtful sampling, model cross-validation designs, and model tuning are necessary to ensure that regression trees fit the data in a way which generalizes well to new samples. The final predictive method is the artificial neural network.

Artificial neural networks. ANNs are a family of nonparametric models that can approximate a wide range of relationships between input and output data. ANNs work both as classifiers and as regressors. ANNs can fit different data patterns by using linear transformations and combinations to create derived features from input data which are then transformed and combined again in each layer of the ANN (Hastie et al., 2009).

To obtain each derived feature Z_m, a dot product is calculated between a single vector of input data X, and a unique vector of randomly initialized weights α_m^T for each combination of X_j and Z_m before being added to an intercept (or bias) term α_{0m}

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T \cdot X). \tag{11}$$

Next, the resultant scaler v is transformed through an activation function $\sigma(v)$ to arrive at Z_m. A convenient activation function is the logistic (or sigmoid) function

$$\sigma(v) = \frac{1}{1 + e^{-v}}. \tag{12}$$

The choice of activation function is a hyperparameter of ANNs. The logistic function is flexible since Z_m can achieve various levels of nonlinearity depending on the value of v_m. The entire vector of derived features Z comprises a hidden layer of an ANN. In a single layer ANN there is only one hidden layer. The size of M and the number of hidden layers are other hyperparameters of ANNs which can be tuned during model training.

To obtain an output T_k, $k = 1, \ldots, K$, the values of the preceding layer are multiplied by a second set of randomly initialized weights β_k^T and intercepts β_{0k}

$$T_k = \beta_{0k} + \beta_k^T \cdot Z. \tag{13}$$

The vector of outputs T is then transformed using an output function $g_k(T)$

$$f_k(X) = g_k(T). \tag{14}$$

If an ANN is being used for a univariate multiple regression task like IDM, $K = 1$ and $g_k(T) = T_k$. If an ANN is being used for classification, K is equal to the total number of classes. In classification, Y is a vector of K elements in which every value is 0 except the value that corresponds to the correct class for an observation (which is 1). A convenient output function for classification is the softmax function

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}} \tag{15}$$

which produces a vector of values between 0 and 1. The values output by the softmax function also sum to 1, a property that is useful for error calculations when comparing against a one-hot encoded vector of training outputs. Applying one-hot encoding to a categorical variable is coding each category as a new variable coded 1 or 0. ANNs structured for classification tasks are those used by distributed representations of language.

Knowledge of the process for estimating the weights in an ANN is also useful to more fully understand distributed representations. The weights ϑ[2]

$$\vartheta = \begin{matrix} \{\alpha_{0m},\ \alpha_m; m = 1, 2, \dots, M\}\ M(p+1), \\ \{\beta_{0k},\ \beta_k; k = 1, 2, \dots, K\}\ K(M+1) \end{matrix} \tag{16}$$

[2] A variant of the Greek letter *theta* is used here since θ is used in (Hastie et al., 2009), but is traditionally used in psychometrics work to represent a person's score on the latent trait.

are estimated by the method of gradient descent through the back-propagation of errors given by the error function (back-propagation, Rumelhart, Hinton, & Williams, 1986). In regression the error function $R(\vartheta)$ is squared error

$$R(\vartheta) = \sum_{i=1}^{N} R_i = \sum_{i=1}^{N} \sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2. \quad (17)$$

A gradient is the derivative of a multivariate space and gives the direction of steepest slope as a vector of partial derivatives with respect to the weights of each layer. Thus, the gradient gives the direction in which the greatest reduction in error can be achieved. Back-propagation uses the chain rule for computing the gradient of the error function for each unit of the ANN independently and simultaneously. Gradient descent by back-propagation is like the Newton-Raphson-style algorithms commonly used in estimation of IRT parameter estimates (Baker & Kim, 2004). An important difference being that gradient descent only uses the first derivative of an error function instead of the second derivative as Newton-Raphson style methods do. Avoiding second derivative calculations can make gradient descent style methods relatively computationally simpler than Newton-Raphson style estimation methods. Back-propagation can iteratively adjust ϑ until $R(\vartheta)$ equals zero. To avoid finding a local minimum of $R(\vartheta)$ that is unique to the training sample, the number of iterations can be tuned as another hyperparameter.

Literature Summary

The present study seeks to bolster the research literature on automated IDM by applying methods novel to automated IDM to a large certification program's item bank in

an operationally realistic yet well-controlled manner. Each of the sections of the literature review has presented lessons and tools which inform the present study. Recent studies in the automated IDM literature provide a scaffold on which to build an operational study design. Recent advances in representing language provide novel methods for automatically extracting item features to serve as the inputs to predictive models. Predictive models with tunable hyperparameters require strict data partitioning design for model cross-validation. Different predictive models come with a variety of features, making some better suited for detecting signal in the item difficulty feature space than others. The idiosyncrasies of the predictive methods motivates a comparative approach.

Standard 4.9 of the *Standards for Educational and Psychological Testing* makes explicit the importance of using representative samples when piloting test items or forms (American Educational Research Association, American Psychological Association, & Joint Committee on Standards for Educational and Psychological Testing, p. 88, 2014). If researchers can extract meaningful item representations and train models to accurately predict item difficulty, then the predictive models would presumably reflect the behavior of the target population. A well-trained predictive model would circumvent the issues introduced by using small pilot samples to estimate how the target population would behave.

Research Questions

The present study was designed to answer the following three research questions:

1. What evidence can be compiled that TF-IDF weighted FastText SG representations meaningfully represent multiple choice item components?

2. Which of the five predictive methods (LLS regression, PC regression, PLS regression, CART, and ANNs) most accurately predicts item difficulty?
3. Can the most accurate predictive model from Research Question 2 provide insight into how item components contribute to item difficulty?

CHAPTER III

METHODS

This section outlines the methods which are designed to answer the research questions of the present study. First, a section describes the item bank that served as the data for the subsequent analyses. Next, the methods planned to answer Research Question 1 (R1) are described. To answer R1, the items are converted to language representations that are then evaluated using a multi-class similarity comparison. The following section explores Research Question 2 (R2) and is geared around building and comparing models which predict item difficulty. Finally, the method for exploring Research Question 3 (R3) is described.

Item Bank Description

The items used in the present study came from the item bank of the ABP GP and GP-MOC examinations (American Board of Pediatrics, 2018). The items were securely administered on initial certification and/or maintenance of certification examinations from 1999 to 2017. The initial certification and maintenance of certification examinations share some items, follow the same test blueprint (see Appendix B, American Board of Pediatrics, 2018), are administered to the same overall population (i.e., general pediatricians at multiple points in their careers), and are written to tap the same latent trait (i.e., knowledge required for safe and effective clinical practice as a general

pediatrician)[3]. Each item is MC format comprised of a question stem and a series of either four or five answer choices. An example item is included in Appendix A (American Board of Pediatrics, 2018). The study sample consists of 4,784 items that have been administered to anywhere between 450 and 28,824 respondents.

Descriptive statistics were calculated on the item bank stems, keys, and aggregated distractors. The item stem word counts ranged from 5 to 306 words (M = 63.71, SD = 37.19). The item stem word counts were slightly positively skewed at 1.34 and presented moderately high kurtosis at 3.16. The item key word counts ranged from 1 to 28 words (M = 4.02, SD = 3.27). The item key word counts were more positively skewed and kurtotic than the stems word counts with a skewness of 1.99 and a kurtosis of 5.22. The aggregated item distractor word counts ranged from 3 to 104 words (M = 15.58, SD = 11.74). The aggregated distractor word counts were even more positively skewed and kurtotic than the item key word counts having a skewness of 2.12 and a kurtosis of 6.13. Of the 4,783 total items, 4,415 (92.31%) had 5 answer choices. The remaining 368 (7.69%) items had 4 answer choices.

---

[3] Although the CERT and MOC-G exams have always been comprised of items that are sampled from the same content domains, there have been points in time where the content domain weights within the blueprint have differed slightly for the CERT and MOC-G exams. Largely speaking, however, and for the purposes of this study, it is accurate to claim that these two exams follow the same test blueprint and are measuring the same overall latent trait.

Each administered test form contained common items with at least one other test from. Common items allowed for all items to be concurrently calibrated using the Rasch model (Rasch, 1961). Rasch model calibrated item difficulty parameters have a few properties that make them a more desirable target on which to train predictive models than item *p*-values. Rasch model item parameters are designed to reflect an item's true location on a linearized latent construct scale. To the extent that items truly tap the latent scale, the points on a scale produced by Rasch model calibration have meaningful differences between them. Also, Rasch model estimated item parameters are independent of the sample that was used to estimate them. The concurrent calibration conducted on the study sample produced item difficulties which more or less lie on a single latent dimension despite having been administered to different samples of respondents. Overall, the item difficulties ranged from -6.04 to 5.37 (M = 0.00, SD = 1.27). The item difficulties were slightly negatively skewed with a skewness of -0.48 and presented approximately normal levels of kurtosis at 1.13.

Generating Word Representations

Each item's text was converted into distributed representations using a pre-trained FastText SG model (Bojanowski et al., 2017). A specialized data preparation process took place to prepare the items for generating representations. All Arabic and Roman numerals in the items were converted to their equivalent text. All punctuation was also removed from the items. All uppercase item characters were converted to lowercase characters. Symbols like "@" and "&" were replaced with their respective text

equivalents (“at” and “and”). After preparation, a 300-dimensional FastText SG word representation was recovered for each word within each item component.

Representation Weighting and Combination

The next step to prepare the items for analysis was to create weighted item representations for each item in the item bank using Equation 2.17. To create weighted representations of each item, first TF-IDF weights were computed for each word in the item bank. Then, each word’s FastText SG representation was multiplied by its corresponding TF-IDF weight. Finally, the TF-IDF weighted word representations for any item component with multiple words was averaged to create a single representation for the stem, key, and each distractor of each item respectively. For example, a multi-sentence item stem is converted to a 300-dimensional item representation vector by multiplying each 300-dimensional word representation vector by its corresponding TF-IDF weight, then taking the average of all of the 300-dimensional weighted word representation vectors.

The distributed representation predictors for each item were the item’s weighted representation of the stem, the weighted representation for the key, and a weighted average representation of the distractors. The distractors were averaged because using a 300-dimensional representation for each component (900 total dimensions) pushes the limits of the predictive methods that need at least as many observations as variables to arrive at stable prediction weights. Another, more pragmatic, reason for combing the distractors into a single representation is that some items have four distractors while

others have three. Using a representation for each distractor would leave missing data for any item with only three distractors.

R1 Methods

The goal of R1 was to evaluate the quality of the weighted item representations. Research literature supports evaluating sentence representations using classification tasks. While earlier work showed promise on a similar task (Rehurek & Sojka, 2010), Gershman & Tenenbaum (2015) found that no representation models could approximate the human similarity rankings in a phrase similarity task. However, in a multi-class classification study using representations of paraphrased sentences with the hypothesis that paraphrased versions of sentences will be classified together, summed and averaged Word2Vec representations outperformed all other models tested (White et al., 2015). In sentence representation research, multi-class classification is the use of sentence representations to classify sentences into multiple known groups. Recently, multi-class classification tasks have found their place in the sentence representation literature as a common sentence representation model evaluation tool (Conneau & Kiela, 2018; Conneau et al., 2018; Perone, Silveira, & Paula, 2018).

In the present research, the 25 item blueprint categories of the GP certification examination from which the item bank came presented an opportunity to implement a unique human-machine multi-class classification similarity comparison task. This study compared in-category item stem and key representations to out-of-category item stem and key representations. The comparison was done by first calculating pairwise cosine similarities between each item/key pair in the item bank. Then, an average cosine

similarity matrix was constructed. The average cosine similarity matrix was organized by test blueprint content domain (American Board of Pediatrics, 2018). The diagonal elements of the cosine similarity matrix are the average of the pairwise cosine similarities for each pair of items in a domain. The off-diagonal elements contain the average of every pairwise combination of two domains. The greater the number of domains that are on average most similar within-domain, the more likely the TF-IDF weighted item representations are capturing the specific idiosyncrasies of the domains. While only a rough measure, to the extent that the 25 blueprint categories reflect item groupings, the procedure should provide reasonable information about how well the TF-IDF weighted FastText SG item representations capture the meaning of the items.

R2 Methods

The present study took a three-step approach to answer R2. First, the item representation data was partitioned into a training, validation, and a test set. Second, each of the five predictive methods outlined in the literature review (LLS regression, PC regression, PLS regression, CART, and ANNs) was trained and validated. Finally, each trained predictive method was evaluated on item difficulty prediction accuracy. After a description of the additional predictor variables, each of the three steps is expanded upon in this section.

Additional predictors. In addition to the distributed item representations, five predictor variables were included in the predictive models. Three of the additional variables were the number of words that made up each item component (i.e., stem, key, and aggregated distractor word counts). Continuous word count variables for each item

component were included to serve as a measure of how much information within each item component was being smoothed over by aggregating the components. Similarly, a binary variable coded 0/1 was included to indicate which items contained five total options (coded 1) and which items contained four total options (coded 0). Lastly, the blueprint categories were included in the prediction models as categorical predictors (coded 0-25) to allow the prediction models to account for any variance attributable to blueprint categories.

Data partitioning. Next, the data set was split into training, validation, and a test partitions. The training, validation, and test sets for the present study were comprised of 70%, 15%, and 15% of the items respectively. A 70%/15%/15% partition keeps the training set large while allowing for the prediction of more than a form's worth of items in each the validation and test sets. A large training set gives the predictive models a better chance of characterizing whatever signal exists in the data. Another motivation for a 70%/15%/15% data partition is that applied researchers would have most of the items in an item bank to train predictive models. Researchers in an operational setting would need to predict a small number of items' difficulties; perhaps only for new items, or only for new test forms. This partition structure should ensure that the validation and test sets are large enough to be representative of the latent trait the items are designed to measure.

The training, validation, and test partitions were sampled from the entire item pool using a stratified sampling design. The strata mirrored the content categories of the test blueprint. Partitioning the item bank based on test blueprint categories increased the probability that each partition is representative of the latent trait the items are collectively

designed to measure. Sampling based on the structure of an operational test form is a consideration which should improve the operational generalizability of the present study. Note however, that the proportions of items in each stratum matches those of the item bank and are unlikely to match the exact proportions outlined in a test blueprint.

Model building. After the item data are partitioned, an iterative process was used to tune the hyperparameters of each of the predictive methods described in the predictive modeling section of the literature review. Apart from LLS regression, each predictive method has its own set of hyperparameters to tune:

- No tuning for LLS regression.
- The number of retained principal components for PC regression.
- The number of retained PLS directions of the PLS regression.
- The number of nodes and splits for CART.
- The number of hidden units in the hidden layer of an ANN.

To start the tuning process for each model, an initial version of a model was fitted to the training data. A hyperparameter of the first version of model was then manually adjusted. The newly adjusted version of the initial model was then compared to the initial model on recovery of the item difficulties of the validation partition. Accurate recovery of validation set item difficulties was assessed by using root-mean-square error (RMSE):

$$RMSE = \sqrt{\frac{\sum(\hat{f}(x_i) - y_i)^2}{N}} \tag{18}$$

Where $\hat{f}(x_i)$ and y_i are the prediction and true value respectively for record i, and N is the number of records in the set. The process of fitting, tuning, and comparison continued until the RMSE of the predictions on the validation partition were minimized. Tuning was repeated for each of the four tunable methods until a final version of each method was selected for comparison against the final versions of the other methods.

Final model comparison. After each model's hyperparameters are appropriately tuned using the training and validation item sets, the models were each compared on their ability to recover the item difficulties of the items in the test set. Note that up to this point the test data set had not been used to train or tune any of the models. The performance of each model on the test set was compared with RMSE and average bias. Average bias was defined as

$$Bias = \frac{\sum(\hat{f}(x_i) - y_i)}{N}. \tag{19}$$

The most accurate model was defined as the model with the lowest RMSE.

R3 Methods

After the model that best predicted item difficulty was found, partial models were created to explore the predictive contribution of each predictor variable. Each of the item component distributed representation vectors was treated as a single predictor. Without changing the final model's hyperparameters, eight partial models were created by excluding a different one of the eight predictor variables. For example, one of the partial-models excluded the 300-dimensional vector of predictors representing the item stem. Another partial model excluded only the item domain predictor.

Then, the RMSE of the complete model was compared to that of each partial model. The difference in RMSE between the complete model and each partial model provided a measure of the relative contribution of each item component to the complete model. The predictor exclusion method was attractive because it accounts for the additional predictive contribution of the interaction effects between the item components yet does not depend on interpretable estimated model weights. For example, the predictive contribution of all the complex interactions related to the 300-dimension vector representing the item key are accounted for when the partial model excluding the key vector is compared to the original model.

CHAPTER IV

RESULTS

Here, the results of the study are presented. The chapter begins with a presentation of descriptive statistics and an assessment of the equivalence of each of the cross-validation sets to the overall item bank. Next, the results of each of research questions are presented. The results of R2 are presented by model and include a description of the tuning procedure for each of the tunable predictive modeling methods.

Item Bank Cross-validation Partition Evaluation

Descriptive statistics were calculated on each of the cross-validation sets to assess the equivalence of the stratified partitioning. Appendix C contains the item counts by domain for each of the cross-validation sets. Appendices D-G contain the overall descriptive statistics for all non-representation predictor variables. For the item difficulty parameters, the stem word count, key word count, and aggregated distractor word count respectively, Appendices C-G each present the domain, *N*-size, mean, *SD*, median, minimum, maximum, range, skew, and kurtosis for each cross-validation set. Appendix H contains the domain, total *N*-size, and count of the number of items with five total options for each cross-validation set. The cross-validation sets did not present any concerning departures from the overall item bank on item component word counts, the proportion of items with five answer choices, nor item difficulties.

R1 Results

Appendix I is the lower triangle of the average pairwise cosine similarity matrix for each of the active domains (1-25). The pairwise cosine similarity matrix does not include obsolete items. The diagonal of Appendix I shows the average within-domain pairwise cosine similarity. Each cell of the off-diagonal of Appendix I shows the average pairwise cosine similarity between items in the two corresponding domains. The cosine similarities in Appendix I range from .374 to .534. Overall, the average within-domain cosine similarity was .467, while the average cosine similarity for items in different domains was .443.

Table 1 lists the domain with which each domain had the highest average cosine similarity. Items within 14 of the 25 domains were more similar, on average, to one another than to the items of any other domain. For example, domains 2 (Fetal and Neonatal Care), 5 (Mental and Behavioral Health), and 8 (Infectious Diseases) had the highest average cosine similarity within-domain rather than with any other domain. Domains 11 (Allergy and Immunology), 21 (Urology and Genital Disorders), and 24 (Ethics) were the most general domains; each being most similar within-domain and with three other domains. For example, domain 11 was the most similar domain for domain 4 (Genetics, Dysmorphology, and Metabolic Disorders), 10 (Hematology), 20 (Nephrology, Fluids, and Electrolytes), and within-group.

While the results of R1 are encouraging, readers should take care not to overinterpret the top match results. The magnitudes of the differences of average cosine similarities across all comparisons are small. The range of all average cosine similarities

is only .197. In many cases when the top match is within-domain, the differences between the within-group top match and the next highest matches are small. For example, domain 5 was most similar within-group, having a within-group average cosine similarity of .446. However, the next most similar domain for domain 5 was domain 24 within an average cosine similarity of .445. While calculating the exact probability of the pattern of the cosine similarity task is beyond the scope of the present study, it is below chance levels despite the low magnitudes of the average similarity differences.

Table 1. Most Similar Content Domain.

Domain	Top Match
1	24
2	2*
3	21
4	11
5	5*
6	2
7	21
8	8*
9	9*
10	11
11	11*
12	12*
13	13*
14	14*
15	14
16	16*
17	17*
18	21
19	19*
20	11
21	21*
22	22*
23	24
24	24*

25	24

Note. * Indicates content domains that are most similar within-domain.

R2 Results

The predictive modeling process was conducted using SAS Enterprise Miner 14.2 (SAS Institute Inc., 2016). Each predictive method was fit to the training data and, for all but LLS regression, tuned to reduce training data overfit. The tuning process for each of the four tuned methods is described as laid out in the methods chapter. First, presented below are the prediction results of the LLS regression model. Then, the tuning processes and test set prediction results for each of the tunable predictive methods are outlined. Table 2 contains the final Pearson correlation coefficient, RMSE, and bias values on each cross-validation set for each predictive method.

The R2 results section is brief due to the consistency of findings. No predictive method was able to make predictions on the test set with RMSE values smaller than the 1.287 standard deviation of the item difficulties of test set. In short, no method provided prediction above chance levels. Also, since the elements of the distributed representation vectors are not interpretable and there was no predictive power overall, no model coefficients are reported.

Table 2. R2 Model Results Across Cross-validation Set

	Training			Validation			Test		
Model	*r*	Bias	RMSE	*r*	Bias	RMSE	*r*	Bias	RMSE
LLS	0.535	0.000	1.269	0.023	-0.020	1.657	0.010	-0.016	1.643
PC	0.032	0.001	1.276	-0.005	-0.005	1.249	0.023	0.001	1.288
PLS	0.082	0.001	1.259	0.010	-0.019	1.263	0.040	-0.002	1.295
CART	NA	0.001	1.276	NA	-0.002	1.249	NA	0.002	1.287
ANN	0.140	-0.021	1.190	0.111	-0.004	1.269	0.021	-0.021	1.341

Note. Correlations are not possible for the CART model because all predicted values are equivalent.

LLS regression. The LLS regression was fit to the training data and used to generate predictions for the validation and test sets. The RMSE of LLS regression on the training set was 1.269. The RMSE of the LLS regression on the validation set was 1.657. As expected, since the LLS regression is not a tunable model, predictions on the test set closely matched the performance of those on the validation set with an RMSE of 1.643. Note that the RMSE of the test set is greater than the SD of the test set item difficulties (1.287). The bias of the LLS predictions on the test set were -0.016.

PC regression. The final PC regression model was created by first using the correlation matrix of the predictor variables to calculate the eigenvalues of the predictor variables. Then, the PC regression was tuned by sequentially changing the number of principal components used in the subsequent regression to minimize the RMSE of the predictions in the validation set. The tuning process was stopped once the RMSE of the validation set predictions hit a floor, then began to raise consistently for several iterations of increasing the number of retained components. Due to the manual nature of the process of saving model validation set performance results, only the details of the best performing version of a model was saved (this is also true for each of the remaining

predictive modeling methods). The best performing PC regression model on the validation set was a model that retained the top 3 principal components that accounted for the most variance in the predictors of the training set. The RMSE of the PC regression on the training set was 1.276. The RMSE of the PC regression on the validation set was 1.249. The test set predictions of the final PC regression had an RMSE of 1.288; nearly exactly matching the standard deviation of the item difficulties of the test set. The bias of the final PC regression model predictions on the test set was 0.001.

PLS regression. Tuning the PLS regression was done by sequentially changing the number of extracted factors used as predictors until the RMSE of the validation set predictions was minimized. The PLS regression tuning process was stopped after the RMSE on the validation set hit a floor then rose consistently for several iterations. The best performing PLS regression model on the validation set was a model that extracted 2 factors from the training set. The RMSE of PLS regression on the training set was 1.259. The RMSE of the PLS regression on the validation set was 1.263. The test set predictions of the final PLS regression had an RMSE of 1.295. The bias of the final PLS regression model predictions on the test set was -0.002.

CART. The best performing CART model on the validation set was a model that was pruned down to a single node when training. Tuning parameters for the CART model were the number of splits and the number of nodes. No changes to the number of splits or nodes changed the structure of the final CART model. The single node of the final CART model predicted the mean of the training set for every record of the validation set. The RMSE of the final CART model on the training set was 1.276. The RMSE of the CART

model on the validation set was 1.249. The test set predictions of the final CART model had an RMSE of 1.287. The bias of the final CART model predictions on the test set was 0.002. The single node CART model was the best performing predictive method overall.

ANN. Tuning the ANN was done by sequentially changing the number of units in the hidden layer until a validation set RMSE floor was reached and consistently unbeaten for several iterations. The best performing ANN architecture on validation set contained 3 hidden units in the hidden layer. The RMSE of the final ANN model on the training set was 1.190. The RMSE of the ANN model on the validation set was 1.269. The test set predictions of the final ANN model had an RMSE of 1.341. Although the magnitudes of the differences in RMSE values across the cross-validation sets was small, the pattern of RMSE values of the ANN predictions is indicative of model overfit. The bias of the final ANN regression model predictions on the test set was 0.002.

R3 Results

The CART method predictions were the predictions with the lowest RMSE. As such, eight sub-models of the CART model were created. Each of the CART sub-models excluded one of the sets of distributed representations or one of the other predictor variables. Because the CART model predicted the mean difficulty of the training set for each of the items in the test set, the results of each of the R3 sub-models were equivalent to the full CART model in R2.

CHAPTER V

DISCUSSION

This section begins with an interpretative summary of the findings related to each research question. In the second part of this section, limitations of the study design are identified. Within the limitations section, each limitation of the present study is followed with a discussion of potential avenues for addressing that limitation in future research.

R1 Discussion

The result that items within 14 of the 25 domains were, on average, more similar within domain than to the items of any other domain is intriguing. This finding suggests that the weighted word representations are capturing some conceptual similarity between items within domains. On the other hand, it is perhaps more likely that the items within any one domain are written in a similar way, or often use similar words. Within the context of this study, using similar words would lead to similar word representations comprising each item. Also, items within a domain being written in a similar way leads to similar TF-IDF weights for the words used within that domain. By extension, the final weighted and averaged representation of an item within a domain would be similar.

The present findings make it difficult to place this study's TF-IDF weighted item representations method on the spectrum from capturing deeper meaning to simple word similarity. However, the method used in this study is relatively simple to implement and doesn't require SME input. As it is, the method used to address R1 may be useful as a

piece of validity evidence supporting the accuracy of the blueprint categorizations by giving test developers an idea about the linguistic similarity of their test items. In any case, the study's findings indicate that the weighted averaged word representations strategy is encoding enough information to draw connections between theoretically similar items. A future study exploring representation item groupings with SMEs is planned.

R2 Discussion

The results of R2 show that the best predictions in the present study were achieved by the CART model which predicted the mean of the training set for each item in the test set. The RMSE of the LLS regression on the training set implies a high degree of overfit. LLS regression may not be the best choice of modeling approach to serve as the baseline of future studies since it performs worse than simply predicting the mean of the training set (discussed in the limitations section). The final PC and PLS regression models also predicted item difficulty parameters around the mean of the training set for the items in the test set, and so nearly matched the results of the CART model. The final ANN model was only able to outperform the LLS regression.

The results of the R2 show that none of the predictive modeling methods were able to characterize any signal in the test set. Each of the final model results indicate that if there were any signal in the distributed representations of items, it was far too weak to be characterized by the predictive models. Overall, the results of R2 suggest that the present study's method of creating item representations did not encode properties of the items that are associated with item difficulty. Despite the lack of predictive power, the

findings of R2 can be a stepping stone to future research. The distributed representations method may prove successful in other contexts, or at a minimum, help researchers inform their decisions when choosing representation methods in future studies.

R3 Discussion

Unfortunately, the CART method did not lend itself to producing any additional information about the predictors in the R3 task. Just as the original CART model, the CART sub-models each only produced a single node that predicted the mean of the training set for every item in the test set. The R3 finding further drives home the lack of detectable signal in the training set.

Connections to Existing Research

The goal of this section is to place this study within the context of the research literature outlined in Chapter 2. The present study is situated within context of the broader automated IDM literature by drawing links to and highlighting its differences from the motivating literature.

One way the present study differentiated itself from other studies is how it used word count as a predictor variable. Like Rupp et al. (2001), this study used word count as a predictor variable. However, the present study calculated word counts for each of the components instead of overall for each item. This present study was not able to match the Rupp et al. (2001) study's 31% variability explained. In this study, word counts were only included as a proxy for the information the weighted distributed representations were averaging together. In the professional certification context of the present study it

would have been surprising to find that word count would have had a large influence on the difficulty of the items.

Another important decision made in this study relative to others is its treatment of distractor choices. As done by each of the automated IDM studies, the present study incorporated item distractors. However, this study aggregated the distractors and relied on the predictive modeling methods to identify interactions and main effects between item stems, keys, and distractors. While adding these features was a conceptual improvement, it did not contribute to increased explanation of item difficulty.

The modeling decisions of the present study were also inspired by, but slightly different than those of other researchers. The present study used some of the predictive methods use in Loukina et al. (2016), McLeod et al. (2015) and Rupp et al. (2001) (e.g., LLS regression, and CART). However, the present study did not train LASSO regression, elastic net regression, *k*-nearest neighbors regression, stochastic gradient descent regression, linear or non-linear support vector regressions, or random forest regression. The present study did include PC regression, PLS regression, and ANNs which are predictive methods novel to the automated IDM literature. Like the Loukina et al. (2016) study, the present study also made methodological decisions with respect to modeling that differentiate it from the McLeod et al. (2015) and Rupp et al. (2001) automated IDM studies by including a validation set data partition. The inclusion of a validation set data partition is an important method for reducing the potential for overfit introduced by model tuning procedures. Unlike each Loukina et al. (2016), McLeod et al. (2015) and Rupp et al. (2001) studies, the present study used RMSE as the error metric by which to

compare competing models. Within the context of a large-scale testing program, item difficulty predictions must be accurate and precise to hope to influence the pilot testing process. However, other studies in this area explained some of the variability in item difficulties while the present study did not.

The primary way in which this study differed from other automated IDM studies concerns its generation of the predictor set. The present study generated a large number of predictors somewhat similar to Loukina et al., (2016) and McLeod et al. (2015). However, the primary predictor variables in this study were not specifically selected text complexity features, but weighted and averaged word representations from a pre-trained word representation model. The use of word representations was a novel, although not fruitful, methodological choice in the automated IDM literature.

Limitations and Future Directions

Each limitation is discussed below and followed by suggestions for future studies.

Choice of baseline. This study used LLS regression as a baseline to avoid the reliance on SMEs. The conceptual understanding that SMEs have could mean that their best guess of an item's difficulty is the most reasonable standard against which to compare automated IDM methods. In the case of the present study, the LLS regression baseline performed worse than chance on the test set by overfitting the training data. Future studies may find retrieving predictions from SME's more helpful than LLS regression as a baseline for comparison of predictive methodologies. Specifically, a study using the average predictions of a few SMEs could provide valuable conceptual context for the performance of predictive modeling approaches. Researchers may also be able to

further provide context for their findings by choosing to use the hands-on approaches of the studies outlined in Chapter 2 as meaningful baselines against which to compare automated IDM methods in future studies.

Sample size. One limitation of the present study is the size of the item bank used relative to the total number of predictor variables. By extension, the size of the item bank limited the size of the training set relative to the total number of predictor variables. Training models that have 905 predictors with 3,346 items translates to less than 4 items per predictor variable. Future studies using similar methods to generate predictor variables may increase the probability that any signal that exists within item representations is detected by using a larger item bank. However, signal must be present in the first place for it to be detectable. Other features of this study's design may have decreased the variance within the elements of the FastText word representations.

Averaging and aggregation. Although the words in the predictors were weighted by an importance measure, they are composed by simple averaging. In this study there was a large difference between the language unit of analysis, which were often sentences, and the language units of the representations, which were words or subwords. Even if the original FastText word representations indeed encoded the meaning of the words, the final item stem, key, and distractor representations smoothed over whatever variance existed within those representations. The averaging resulted in many highly kurtotic representation elements centered at zero. Essentially, averaging the representation vectors of many words in high dimensional space may have pulled the item component representations to the origin of the space.

Averaging out the variance in the representations was further exacerbated by aggregating all the distractors of each item as if they belonged to a single sentence. Many of the items also contained clinical vignettes preceding the question stem (see the sample item in Appendix A, American Board of Pediatrics, 2018). Each of the clinical vignettes was aggregated as part of the question stem. Applying TF-IDF weighting was the only defense against the issues brought about by averaging and aggregation. The choice to use TF-IDF as a word representation weighting mechanism in this study may not have been enough to overcome the signal loss issues caused by averaging and aggregation.

Future studies may choose to test other word representation weighting methods (e.g., see Arora et al., 2017). Another approach is to treat each distractor as its own set of predictors. Future studies may also choose to split the clinical vignettes from the question stem before training predictive models. Finally, summing the distributed representations of words within the items may have mitigated the decrease in representation element variance issues introduced by averaging. Limited research comparing summing and averaging of word representations into sentence representations have found that the two approaches perform similarly (White et al., 2015). Future IDM studies may find that summing word representations may increase representation variance in high dimensional space such that predictive models are better able to identify patterns between the representation elements and item difficulties.

Large number of predictors. The decision to use a pre-trained FastText model with 300-dimensional word vectors placed the study in a situation in which 300 was the minimum number of predictors (i.e., if every component of an item were to have been

aggregated). The large number of predictors drove the demand for more training data, and the need for item component aggregation. One approach to work around high numbers of predictors would be to choose different predictive modeling methods. Some modeling methods are designed to handle low sample size to predictor ratios. For example, an elastic-net regression subsets predictor variables and penalizes the parameter weights of correlated predictors (Zou & Hastie, 2005). However, addressing the number of predictors at the source may bare more fruit.

Future studies using FastText may attempt to mitigate the pressure placed on predictive modeling methods in this study by directly re-training FastText to generate representations with fewer elements. As mentioned in the section on distributed representations of language, there is room to adjust the size of the hidden layer and subsequent dimensional length of the word representation when training a distributed language representation model. The early work on Word2Vec used word vector dimensionality as a condition in one of its experiments. Likewise, the work on FastText in a medical language context trained a FastText model with only 100 dimensions. Because representation dimension size may be context specific, future studies may find success by training distributed language representation models with context specific text and compare different hidden layer sizes. Training FastText with context specific text would capture more of the words used in a context; decreasing the reliance on morpheme composition. Context specific training would also limit the training only to the specific word usages within that context (i.e., limiting the number of senses needing to be captured by a single word's representation). Training FastText only with text that

matched the specific testing context may further increase the quality of the resultant word representations and allow researchers to reduce representation length. In effect, context specific training could concentrate and specify the information encoded in word representations.

Language modeling. As stated above, the present study's context included items comprised of many words. Using a language model that functions at the word, or subword, level for IDM in this study's context may have been too granular a task for FastText. A better approach may have been to choose language modeling techniques that work at higher units of language. While the research is in its earlier stages, there are techniques that operate at the sentence level (Cer et al., 2018; Conneau, Kiela, Schwenk, Barrault, & Bordes, 2017). Sentence level models may be more appropriate for MC items with more words in the item components. Like FastText, the sentence level models are based on ANN architecture. However, the sentence level models take complete sentences as inputs. Using sentence level models to encode item components may be a promising direction for future IDM studies.

Item quality. In this study, item difficulty was the only target variable for the predictive methods. Future studies may choose to predict other item statistics. For example, applied researchers may find it useful to attempt to predict item quality indicators used in the key validation and item review processes. Key validation is the process by which exam developers verify that the keys of MC exam items are statistically supported. Item review is a process during which every item's quality statistics are reviewed.

In some situations, test developers may find the ability to model item quality as important as modeling item difficulty. Just as modeling item difficulty may help researchers understand how characteristics of items drive item difficulty, modeling item quality may give test developers the ability to learn what makes high- or low-quality items have the statistical properties that they have. Understanding the properties of items is the goal of principled assessment designs, and while they promote an *a priori* approach to uncovering drivers of item statistical characteristics, exploratory language modeling efforts may uncover surprising and useful relationships. In the distant future practitioners may be able to build intelligent systems which automatically produce, predict the statistical characteristics of, serve, and score items seamlessly.

REFERENCES

American Board of Pediatrics. (2018). General Pediatrics Content Outline. Retrieved February 26, 2019, from https://www.abp.org/content/content-outlines-0

American Educational Research Association, American Psychological Association, & Joint Committee on Standards for Educational and Psychological Testing. (2014). *Standards for educational and psychological testing* (4th ed.). Washington, D.C.: American Educational Research Association.

Ames, A., & Smith, E. (2018). Subjective priors for item response models: Application of elicitation by design. *Journal of Educational Measurement Fall*, *55*(3), 373–402. https://doi.org/10.1111/(ISSN)1745-3984

Arora, S., Liang, Y., & Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of ICLR 2017* (pp. 1–16).

Baker, F. B., & Kim, S.-H. (2004). *Item response theory: Parameter estimation techniques*. New York: Marcel Dekker.

Becker, K., & Masters, J. (2017). Automated modeling of item difficulty on admissions tests. In *Annual Meeting of the National Council on Measurement in Education*. San Antonio, TX.

Bejar, I. I. (1983). Subject matter experts' assessment of item statistics. *Applied Psychological Measurement*, *7*(3), 303–310. https://doi.org/10.1177/014662168300700306

Bodenreider, O. (2004). The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, *32*(90001), D267–D270. https://doi.org/10.1093/nar/gkh061

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. Retrieved from https://www.transacl.org/ojs/index.php/tacl/article/view/999

Botha, J. A., & Blunsom, P. (2014). Compositional morphology for word representations and language modelling. In *Proceedings of the 31st International Conference on Machine Learning*. Retrieved from http://arxiv.org/abs/1405.4273

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. *The Wadsworth statistics/probability series* (Vol. 19).

Camacho-Collados, J., & Pilehvar, M. T. (2018). From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, *63*, 743–788. https://doi.org/10.1002/cne.22311

Cattell, R. B. (1966). The scree test for the numbers of factors. *Multivariate Behavioral Research*, *1*(October), 245–276. https://doi.org/10.1207/s15327906mbr0102

Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. St., … Kurzweil, R. (2018). Universal sentence encoder. Retrieved from http://arxiv.org/abs/1803.11175

Conneau, A., & Kiela, D. (2018). SentEval: An evaluation toolkit for universal sentence representations. Retrieved from http://arxiv.org/abs/1803.05449

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. https://doi.org/10.1.1.156.2685

Conneau, A., Lample, G., & Baroni, M. (2018). *What you can cram into a single vector: Probing sentence embeddings for linguistic properties*. Retrieved from https://arxiv.org/pdf/1805.01070.pdf

Daniel, R. C., & Embretson, S. E. (2010). Designing cognitive complexity in mathematical problem-solving items. *Applied Psychological Measurement*, *34*(5), 348–364. https://doi.org/10.1177/0146621609349801

Devroye, L. P., & Wagner, T. J. (1979). Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, *25*(5), 601–604. https://doi.org/10.1109/TIT.1979.1056087

Embretson, S. E. (1998). A cognitive design system approach to generating valid tests: Application to abstract reasoning. *Psychological Methods*, *3*(3), 380–396. https://doi.org/10.1037//1082-989X.3.3.380

Enright, M. K., Morley, M., & Sheehan, K. M. (2002). Items by design: The impact of systematic feature variation on item statistical characteristics. *Applied Measurement in Education*, *15*(1), 49–74. https://doi.org/10.1207/S15324818AME1501

Ezeiza Alvarez, J. (2017). *A review of word embedding and document similarity algorithms applied to academic text*. Retrieved from http://ad-publications.informatik.uni-freiburg.de/theses/Bachelor_Jon_Ezeiza_2017.pdf

Frank, I. E., & Friedman, J. H. (1993). A statistical view of some chemometrics regression tools. *Technometrics*, *35*(2), 109–135. https://doi.org/10.2307/1269656

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, *19*(1), 1–67. Retrieved from http://www.jstor.org/stable/2241837

Gershman, S. J., & Tenenbaum, J. B. (2015). Phrase similarity in humans and machines. *Proceedings of the 37th Annual Conference of the Cognitive Science Society*, 776–781.

Gierl, M. J., & Haladyna, T. M. (2013). *Automatic item generation: Theory and practice*. New York, NY: Routledge.

Gierl, M. J., & Lai, H. (2016). Automatic item generation. In S. Lane, M. R. Raymond, & T. M. Haladyna (Eds.), *Handbook of Test Development* (2nd ed., pp. 410–430). New York, NY: Routledge.

Gorin, J. S., & Embretson, S. E. (2013). Using cognitive psychology to generate items and predict item characteristics. In *Automatic Item Generation: Theory and Practice* (pp. 136–156).

Hadi, A. S., & Ling, R. F. (1998). Some cautionary notes on the use of principal components regression. *The American Statistician*, *52*(1), 15–19. https://doi.org/10.1080/00031305.1998.10480530

Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2–3), 146–162. https://doi.org/10.1080/00437956.1954.11659520

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. *Springer series in statistics* (2nd ed.). New York, NY: Springer. https://doi.org/10.1007/978-0-387-84858-7

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. *Parallel Distributed Processing*. https://doi.org/10.1146/annurev-psych-120710-100344

Hoshino, A., & Nakagawa, H. (2010). Predicting the difficulty of multiple-choice close questions for computer-adaptive testing. *Research in Computing Science*, *46*(March), 279–292. Retrieved from http://cicling-org.g-sidorov.org/2010/Vol46.pdf#page=289

Huff, K. L. (2003). *An item modeling approach to descriptive score reports*.

Impara, J. C., & Plake, B. S. (1998). Teachers' ability to estimate item difficulty: A test of the assumptions in the Angoff standard setting method. *Journal of Educational Measurement, 35*(1), 69–81. https://doi.org/10.1111/j.1745-3984.1998.tb00528.x

Jolliffe, I. T. (1982). A note on the use of principal components in regression. *Applied Statistics, 31*(3), 300–303. https://doi.org/10.2307/2348005

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation, 28*(1), 11–21.

Jones, K. S. (1973). Index term weighting. *Information Storage and Retrieval, 9*(11), 619–633. https://doi.org/10.1016/0020-0271(73)90043-0

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 427–431). https://doi.org/1511.09249v1

Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning* (Vol. 32, pp. 1188–1196). https://doi.org/10.1145/2740908.2742760

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge University Press. https://doi.org/10.1017/CBO9781139924801

Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27* (pp. 2177–2185). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf

Lorge, I., & Kruglov, L. (1953). The improvement of estimates of test difficulty. *Educational and Psychological Measurement*, *13*(1), 34–46.

Loukina, A., Yoon, S., Sakano, J., Wei, Y., & Sheehan, K. M. (2016). Textual complexity as a predictor of difficulty of listening items in language proficiency tests. *Proceedings of the 26th International Conference on Computational Linguistics (COLING-16)*, 3245–3253.

Luecht, R. (2013). Assessment engineering task model maps, task models and templates as a new way to develop and implement test specifications. *Journal of Applied Testing Tec*, *14*, 1–28.

Luong, M.-T., Socher, R., & Manning, C. D. (2013). Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning* (pp. 104–113).

McLeod, J., Butterbaugh, D., Masters, M., & Schaper, E. (2015). Predicting item difficulty by analysis of language features. In *Paper presented at the 2015 Annual Meeting for the National Council on Measurement in Education*. Chicago, IL.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *Arxiv*, 1–12. https://doi.org/10.1162/153244303322533223

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the Language Resources Evaluation Conference*. Retrieved from http://arxiv.org/abs/1712.09405

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of Neural Information Processing Systems 26* (pp. 1–9). https://doi.org/10.1162/jmlr.2003.3.4-5.951

Mislevy, R. J., Almond, R. G., & Lukas, J. F. (2003). *A brief introduction to evidence-centered design*. Princeton, NJ. Retrieved from https://mstu-commons.wikischolars.columbia.edu/file/view/RR-03-16.pdf

Mislevy, R. J., Sheehan, K. M., & Wingersky, M. (1993). How to equate tests with little or no data. *Journal of Educational Measurement*, *30*(1), 55–78. Retrieved from http://www.jstor.org/stable/1435164

Perone, C. S., Silveira, R., & Paula, T. S. (2018). *Evaluation of sentence embeddings in downstream and linguistic probing tasks*. Retrieved from http://arxiv.org/abs/1806.06259

Python Software Foundation. (2017). Python Language Reference, version 2.6.2. *Python Software Foundation*. https://doi.org/https://www.python.org/

Qiu, S., Cui, Q., Bian, J., Gao, B., & Liu, T.-Y. (2014). Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (pp. 141–150). Dublin, Ireland: Dublin City University and Association for Computational Linguistics. Retrieved from http://www.aclweb.org/anthology/C14-1015

Rasch, G. (1961). Probabilistic models for some intelligence and attainment tests. *Information and Control*. https://doi.org/10.1016/S0019-9958(61)80061-2

Rehurek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45–50). https://doi.org/10.13140/2.1.2393.1847

Ripley, B. D. (1996). *Pattern recognition and neural networks*. Cambridge University Press.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Rudner, L. (2010). Implementing the Graduate Management Admission Test computerized adaptive test. In W. J. van der Linden & C. A. W. Glas (Eds.), *Elements of Adaptive Testing* (pp. 151–165). New York, NY: Springer.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536. https://doi.org/10.1038/323533a0

Rupp, A. A., Garcia, P., & Jamieson, J. (2001). Combining multiple regression and CART to understand difficulty in second language reading and listening comprehension test items. *International Journal of Testing*, *1*(3–4), 185–216. https://doi.org/Article

SAS Institute Inc. (2016). SAS Enterprise Miner. Cary, NC.

Sheehan, K. M., & Mislevy, R. J. (1994). *A tree-based analysis of items from an assessment of basic mathematics skills*. Princeton, NJ.

Shermis, M. D., & Burstein, J. (2013). *Handbook of automated essay evaluation: Current applications and new directions*. New York, NY: Routledge.

Shermis, M. D., Burstein, J., & Bursky, S. A. (2013). Introduction to automated essay evaluation. In *Handbook of Automated Essay Evaluation: Current Applications and New Directions* (pp. 1–15). https://doi.org/10.4324/9780203122761.ch1

Shermis, M. D., & Hamner, B. (2013). Contrasting state-of-the-art automated scoring of essays. In *Handbook of Automated Essay Evaluation: Current Applications and New Directions* (pp. 313–346). https://doi.org/10.4324/9780203122761.ch19

Soricut, R., & Och, F. J. (2015). Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter*

of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT 2015) (pp. 1626–1636). Denver, CO.

Stone, M., & Brooks, R. J. (1990). Continuum regression: Cross-validated sequentially constructed prediction embracing ordinary least squares, partial least squares and principal components regression. *Royal Statistical Society. Series B (Methodological), 53*(2), 237–269. https://doi.org/10.1016/S0198-0254(06)80522-4

White, L., Togneri, R., Liu, W., & Bennamoun, M. (2015). How well sentence embeddings capture meaning. In *Proceedings of the 20th Australasian Document Computing Symposium* (pp. 1–8). https://doi.org/10.1145/2838931.2838932

White, L., Togneri, R., Liu, W., & Bennamoun, M. (2018). *Finding word sense embeddings of known meaning*.

Wold, H. (1966). Estimation of principal components and related models by iterative least squares. In P. R. Krishnaiah (Ed.), *Multivariate Analysis* (pp. 391–420). New York, NY: Academic Press.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *39*(24), 4076–4084. https://doi.org/10.1111/j.1467-9868.2005.00503.x

APPENDIX A

SAMPLE ITEM

A 12-month-old patient who has been appropriately treated for bacterial meningitis is now being discharged from the hospital.

This child's parents should be informed that the most common sequela of bacterial meningitis is which of the following?

A. Seizures

B. Sensorineural hearing loss

C. Blindness

D. Brain abscess

E. Paralysis

APPENDIX B

GENERAL PEDIATRICS TEST BLUEPRINT CONTENT DOMAINS

Domain
1. Preventive Pediatrics/Well-Child Care
2. Fetal and Neonatal Care
3. Adolescent Care
4. Genetics, Dysmorphology, and Metabolic Disorders
5. Mental and Behavioral Health
6. Child Abuse and Neglect
7. Emergency and Critical Care
8. Infectious Diseases
9. Oncology
10. Hematology
11. Allergy and Immunology
12. Endocrinology
13. Orthopedics and Sports Medicine
14. Rheumatology
15. Neurology
16. Eye, Ear, Nose, and Throat
17. Cardiology
18. Pulmonology
19. Gastroenterology
20. Nephrology, Fluids, and Electrolytes
21. Urology and Genital Disorders
22. Skin/Dermatology
23. Psychosocial Issues
24. Ethics
25. Research Methods, Patient Safety, and Quality Improvement

APPENDIX C

DOMAIN COUNTS AND FREQUENCY BY CROSS-VALIDATION SET

Domain	N	Frequency	Training N	Training Frequency	Validation N	Validation Frequency	Test N	Test Frequency
0	1089	0.228	762	0.228	164	0.228	163	0.227
1	483	0.101	338	0.101	72	0.1	73	0.102
2	203	0.042	142	0.042	30	0.042	31	0.043
3	136	0.028	95	0.028	20	0.028	21	0.029
4	219	0.046	153	0.046	33	0.046	33	0.046
5	182	0.038	127	0.038	28	0.039	27	0.038
6	146	0.031	102	0.03	22	0.031	22	0.031
7	179	0.037	125	0.037	27	0.038	27	0.038
8	250	0.052	175	0.052	38	0.053	37	0.052
9	70	0.015	49	0.015	10	0.014	11	0.015
10	104	0.022	73	0.022	16	0.022	15	0.021
11	104	0.022	73	0.022	16	0.022	15	0.021
12	140	0.029	98	0.029	21	0.029	21	0.029
13	187	0.039	131	0.039	28	0.039	28	0.039
14	74	0.015	52	0.016	11	0.015	11	0.015
15	114	0.024	80	0.024	17	0.024	17	0.024
16	148	0.031	104	0.031	22	0.031	22	0.031
17	148	0.031	104	0.031	22	0.031	22	0.031
18	152	0.032	106	0.032	23	0.032	23	0.032
19	122	0.026	85	0.025	18	0.025	19	0.026
20	119	0.025	83	0.025	18	0.025	18	0.025
21	89	0.019	62	0.019	14	0.019	13	0.018
22	152	0.032	106	0.032	23	0.032	23	0.032
23	36	0.008	25	0.007	6	0.008	5	0.007
24	58	0.012	41	0.012	8	0.011	9	0.013
25	79	0.017	55	0.016	12	0.017	12	0.017

APPENDIX D

ITEM DIFFICULTY DESCRIPTIVE STATISTICS

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Overall	Overall	4783	0.000	1.274	0.075	-6.044	5.368	11.412	-0.475	1.126
Training	Overall	3346	0.000	1.277	0.066	-6.016	4.823	10.839	-0.454	1.084
Validation	Overall	719	-0.002	1.250	0.113	-6.044	3.515	9.559	-0.595	1.454
Test	Overall	718	0.002	1.287	0.062	-5.347	5.368	10.715	-0.462	0.994
Training	0	762	0.026	1.373	0.125	-6.016	4.823	10.839	-0.626	1.432
Validation	0	164	-0.131	1.394	-0.095	-5.244	3.515	8.759	-0.346	0.517
Test	0	163	0.015	1.418	0.130	-4.604	5.368	9.972	-0.109	0.747
Overall	0	1089	0.001	1.383	0.110	-6.016	5.368	11.384	-0.501	1.179
Training	1	338	-0.149	1.271	-0.026	-5.200	3.490	8.690	-0.494	0.891
Validation	1	72	-0.103	1.481	0.171	-5.224	2.885	8.108	-1.157	1.930
Test	1	73	0.061	1.179	0.064	-2.627	2.949	5.576	0.024	-0.673
Overall	1	483	-0.110	1.290	0.009	-5.224	3.490	8.713	-0.596	1.133
Training	2	142	-0.006	1.199	-0.017	-3.854	4.248	8.102	-0.046	1.288
Validation	2	30	-0.070	1.037	-0.077	-1.986	1.783	3.768	-0.304	-0.716
Test	2	31	-0.132	1.022	0.019	-2.397	1.680	4.077	-0.659	-0.252
Overall	2	203	-0.034	1.146	-0.009	-3.854	4.248	8.102	-0.125	1.108
Training	3	95	-0.316	1.223	-0.214	-3.903	2.709	6.612	-0.390	0.287
Validation	3	20	0.183	1.229	0.215	-1.866	2.112	3.977	-0.144	-1.313
Test	3	21	-0.110	1.160	0.014	-3.736	1.564	5.300	-1.202	2.086
Overall	3	136	-0.210	1.219	-0.095	-3.903	2.709	6.612	-0.462	0.369
Training	4	153	0.214	1.092	0.300	-5.025	2.983	8.007	-0.884	3.290
Validation	4	33	0.179	0.669	0.102	-1.499	1.376	2.875	-0.304	-0.436
Test	4	33	0.276	1.442	0.435	-4.570	2.866	7.436	-1.040	1.886
Overall	4	219	0.218	1.097	0.295	-5.025	2.983	8.007	-0.949	3.510
Training	5	127	-0.421	1.191	-0.367	-3.950	2.323	6.272	-0.302	-0.113
Validation	5	28	-0.173	1.402	-0.099	-3.108	3.248	6.355	-0.015	-0.162
Test	5	27	-0.392	1.592	-0.608	-5.347	2.866	8.213	-0.580	1.685
Overall	5	182	-0.378	1.286	-0.302	-5.347	3.248	8.595	-0.315	0.796
Training	6	102	-0.095	1.229	0.047	-3.522	2.369	5.891	-0.493	-0.166
Validation	6	22	-0.145	1.215	-0.143	-2.760	2.146	4.905	0.024	-0.551
Test	6	22	-0.275	1.134	-0.218	-2.876	1.846	4.722	-0.315	0.099
Overall	6	146	-0.129	1.206	-0.044	-3.522	2.369	5.891	-0.389	-0.135
Training	7	125	-0.043	1.252	0.058	-4.796	3.563	8.359	-0.617	1.760
Validation	7	27	0.067	1.037	0.018	-1.695	1.823	3.518	0.175	-1.022
Test	7	27	-0.237	1.607	0.363	-4.443	1.801	6.244	-1.011	0.051
Overall	7	179	-0.056	1.278	0.097	-4.796	3.563	8.359	-0.744	1.534
Training	8	175	0.079	1.199	0.073	-2.943	3.569	6.512	-0.228	0.156
Validation	8	38	0.194	1.032	0.311	-1.732	1.987	3.718	-0.320	-0.935
Test	8	37	0.041	1.194	-0.001	-2.269	3.059	5.328	0.001	-0.257
Overall	8	250	0.091	1.171	0.154	-2.943	3.569	6.512	-0.215	0.060

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Training	9	49	0.184	1.040	0.117	-2.253	2.133	4.386	-0.103	-0.500
Validation	9	10	-0.163	1.011	-0.524	-1.562	1.753	3.314	0.424	-1.145
Test	9	11	-0.169	0.968	0.002	-1.663	1.223	2.887	-0.013	-1.521
Overall	9	70	0.079	1.024	0.023	-2.253	2.133	4.386	0.014	-0.648
Training	10	73	0.010	1.394	0.075	-4.029	3.044	7.073	-0.455	0.174
Validation	10	16	0.184	1.098	-0.369	-1.305	2.169	3.474	0.540	-1.170
Test	10	15	0.338	1.069	0.278	-1.242	2.735	3.977	0.514	-0.485
Overall	10	104	0.084	1.305	0.088	-4.029	3.044	7.073	-0.356	0.376
Training	11	73	0.145	1.016	0.188	-2.704	3.579	6.283	0.077	1.026
Validation	11	16	0.207	0.902	0.445	-2.052	1.230	3.282	-1.191	0.452
Test	11	15	-0.482	1.357	-0.171	-3.198	1.239	4.436	-0.820	-0.508
Overall	11	104	0.064	1.069	0.230	-3.198	3.579	6.776	-0.474	1.347
Training	12	98	-0.084	1.141	0.065	-4.054	2.537	6.590	-0.886	1.343
Validation	12	21	0.336	1.206	0.503	-2.912	1.857	4.769	-0.841	0.229
Test	12	21	-0.139	1.551	-0.065	-4.546	1.837	6.383	-1.095	1.039
Overall	12	140	-0.029	1.220	0.091	-4.546	2.537	7.083	-0.982	1.535
Training	13	131	0.097	1.232	0.159	-4.588	3.098	7.686	-0.756	1.201
Validation	13	28	0.110	1.044	0.282	-2.414	1.669	4.084	-0.721	-0.138
Test	13	28	0.175	0.844	0.194	-1.651	1.691	3.342	-0.293	-0.630
Overall	13	187	0.111	1.150	0.205	-4.588	3.098	7.686	-0.764	1.295
Training	14	52	0.487	1.295	0.608	-2.664	4.391	7.055	-0.095	0.791
Validation	14	11	-0.087	0.712	-0.023	-1.397	1.037	2.434	-0.150	-1.090
Test	14	11	0.386	1.015	0.586	-1.632	2.214	3.846	-0.187	-0.433
Overall	14	74	0.387	1.193	0.503	-2.664	4.391	7.055	0.031	0.998
Training	15	80	-0.173	1.434	0.104	-3.829	2.912	6.741	-0.485	0.275
Validation	15	17	-0.388	1.389	-0.303	-3.012	2.156	5.169	-0.124	-0.905
Test	15	17	-0.012	1.002	-0.158	-2.186	1.758	3.943	-0.216	-0.495
Overall	15	114	-0.181	1.365	-0.075	-3.829	2.912	6.741	-0.451	0.264
Training	16	104	0.088	1.230	0.214	-4.165	2.852	7.017	-0.483	0.921
Validation	16	22	0.119	1.223	0.102	-1.842	3.427	5.269	0.568	0.425
Test	16	22	0.659	1.324	0.915	-2.889	2.185	5.073	-0.998	0.360
Overall	16	148	0.178	1.251	0.241	-4.165	3.427	7.592	-0.391	0.640
Training	17	104	-0.059	1.284	0.031	-5.200	2.117	7.317	-0.942	1.432
Validation	17	22	0.282	0.923	0.401	-1.364	2.222	3.587	0.317	-0.347
Test	17	22	-0.492	1.710	0.073	-3.920	1.686	5.606	-0.534	-1.127
Overall	17	148	-0.072	1.319	0.060	-5.200	2.222	7.423	-0.920	1.067
Training	18	106	0.067	1.269	0.052	-2.674	3.162	5.836	-0.015	-0.569
Validation	18	23	0.314	1.138	0.400	-1.920	3.283	5.203	0.402	0.334
Test	18	23	-0.053	1.129	-0.244	-1.444	2.219	3.663	0.614	-0.823
Overall	18	152	0.086	1.227	0.050	-2.674	3.283	5.957	0.103	-0.410
Training	19	85	0.072	1.245	-0.051	-2.642	4.493	7.135	0.525	0.837
Validation	19	18	0.162	0.955	0.215	-1.884	1.644	3.527	-0.309	-0.470
Test	19	19	0.393	1.293	0.584	-2.148	2.314	4.462	-0.506	-0.916
Overall	19	122	0.135	1.211	0.073	-2.642	4.493	7.135	0.280	0.446
Training	20	83	0.099	1.323	0.173	-3.726	3.255	6.981	-0.394	0.031
Validation	20	18	0.173	1.053	0.073	-1.868	2.181	4.050	0.169	-0.710

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Test	20	18	0.160	0.988	0.173	-1.661	2.057	3.718	-0.035	-0.805
Overall	20	119	0.120	1.232	0.125	-3.726	3.255	6.981	-0.343	0.162
Training	21	62	-0.009	1.475	-0.065	-3.273	4.017	7.290	0.308	0.272
Validation	21	14	-0.863	2.232	-0.392	-6.044	2.335	8.379	-0.615	-0.227
Test	21	13	-0.350	1.021	-0.178	-1.713	1.419	3.132	0.128	-1.368
Overall	21	89	-0.193	1.577	-0.149	-6.044	4.017	10.061	-0.284	1.464
Training	22	106	0.105	1.259	0.078	-3.815	3.518	7.333	-0.186	0.306
Validation	22	23	-0.232	1.094	0.123	-3.131	1.805	4.936	-0.649	0.133
Test	22	23	-0.040	1.034	-0.020	-1.797	1.703	3.500	-0.087	-1.245
Overall	22	152	0.032	1.203	0.078	-3.815	3.518	7.333	-0.193	0.318
Training	23	25	0.257	0.912	0.212	-1.612	2.909	4.520	0.659	1.159
Validation	23	6	-0.026	1.936	0.651	-2.865	2.240	5.105	-0.361	-1.754
Test	23	5	-0.694	0.984	-0.743	-1.792	0.390	2.181	0.019	-2.158
Overall	23	36	0.078	1.151	0.206	-2.865	2.909	5.773	-0.170	0.424
Training	24	41	-0.015	1.323	-0.077	-3.092	2.757	5.849	-0.341	-0.154
Validation	24	8	0.054	1.323	0.148	-2.120	2.028	4.148	-0.168	-1.271
Test	24	9	0.336	0.844	0.360	-1.365	1.531	2.896	-0.521	-0.551
Overall	24	58	0.049	1.249	0.211	-3.092	2.757	5.849	-0.412	-0.017
Training	25	55	0.118	1.378	0.247	-4.792	4.113	8.905	-0.563	2.363
Validation	25	12	0.534	1.358	0.597	-1.348	2.552	3.900	0.036	-1.732
Test	25	12	0.355	1.307	0.396	-1.891	2.877	4.767	0.119	-0.774
Overall	25	79	0.217	1.357	0.277	-4.792	4.113	8.905	-0.405	1.666

APPENDIX E

STEM WORD COUNT DESCRIPTIVE STATISTICS

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Overall	Overall	4783	63.709	37.190	57	5	306	301	1.339	3.164
Training	Overall	3346	63.692	37.256	57	5	306	301	1.362	3.336
Validation	Overall	719	64.057	36.976	58	9	268	259	1.255	2.723
Test	Overall	718	63.437	37.142	57	9	254	245	1.308	2.753
Training	0	762	62.723	35.399	58	8	303	295	1.431	4.889
Validation	0	164	61.860	37.917	54	10	261	251	1.646	4.424
Test	0	163	64.294	36.191	58	11	188	177	0.749	0.098
Overall	0	1089	62.828	35.880	57	8	303	295	1.368	4.083
Training	1	338	53.760	32.052	48	5	170	165	0.788	0.282
Validation	1	72	58.486	34.451	57	10	181	171	0.829	0.765
Test	1	73	51.616	31.079	51	11	160	149	1.111	1.599
Overall	1	483	54.141	32.269	50	5	181	176	0.852	0.595
Training	2	142	64.183	41.576	55	11	232	221	1.440	2.383
Validation	2	30	67.633	47.103	52	9	186	177	1.050	0.188
Test	2	31	47.548	31.832	42	12	151	139	1.214	1.406
Overall	2	203	62.153	41.421	52	9	232	223	1.409	2.143
Training	3	95	59.695	29.551	59	13	146	133	0.368	-0.074
Validation	3	20	63.350	29.853	62	12	125	113	0.169	-0.738
Test	3	21	71.048	36.654	63	24	192	168	1.585	3.036
Overall	3	136	61.985	30.810	61	12	192	180	0.725	1.443
Training	4	153	62.484	34.198	54	12	194	182	1.356	2.510
Validation	4	33	54.818	32.548	54	13	180	167	1.626	4.198
Test	4	33	56.788	33.712	50	15	194	179	2.091	5.978
Overall	4	219	60.470	33.875	52	12	194	182	1.513	3.250
Training	5	127	73.268	49.567	63	10	306	296	1.483	3.515
Validation	5	28	65.571	40.272	57	14	189	175	1.217	1.274
Test	5	27	75.333	42.117	68	12	164	152	0.395	-0.809
Overall	5	182	72.390	47.075	63	10	306	296	1.384	3.199
Training	6	102	82.157	51.330	74	8	255	247	1.234	1.752
Validation	6	22	70.864	34.295	70	14	156	142	0.396	-0.332
Test	6	22	66.136	35.639	56	11	142	131	0.716	-0.473
Overall	6	146	78.041	47.231	71	8	255	247	1.290	2.271
Training	7	125	64.296	32.067	64	10	219	209	1.794	6.282
Validation	7	27	58.704	26.844	51	27	128	101	0.808	-0.247
Test	7	27	69.222	51.177	60	13	196	183	1.177	0.619
Overall	7	179	64.196	34.822	62	10	219	209	1.698	4.709
Training	8	175	66.063	40.771	58	10	225	215	1.118	1.273
Validation	8	38	67.289	48.598	59	14	268	254	1.923	5.312
Test	8	37	70.378	49.690	53	16	218	202	1.431	1.556
Overall	8	250	66.888	43.264	57	10	268	258	1.406	2.622

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Training	9	49	64.327	36.397	56	13	177	164	1.070	1.282
Validation	9	10	62.000	32.345	61	16	112	96	0.065	-1.522
Test	9	11	44.818	33.662	33	9	95	86	0.336	-1.723
Overall	9	70	60.929	35.665	56	9	177	168	0.874	0.979
Training	10	73	68.836	39.360	64	12	262	250	1.878	6.388
Validation	10	16	75.250	41.253	75	11	160	149	0.230	-0.610
Test	10	15	60.933	27.907	57	10	127	117	0.536	0.245
Overall	10	104	68.683	38.106	64	10	262	252	1.579	5.219
Training	11	73	57.904	32.228	52	9	173	164	1.470	3.022
Validation	11	16	79.813	29.762	73	30	134	104	0.300	-1.065
Test	11	15	71.400	36.555	71	16	146	130	0.387	-0.849
Overall	11	104	63.221	33.304	56	9	173	164	1.041	1.227
Training	12	98	77.122	40.130	69	13	203	190	0.976	0.823
Validation	12	21	70.667	28.205	76	30	138	108	0.426	-0.628
Test	12	21	80.762	36.711	76	22	150	128	0.408	-0.809
Overall	12	140	76.700	37.951	70	13	203	190	0.920	0.807
Training	13	131	58.550	33.733	55	11	201	190	1.126	2.121
Validation	13	28	59.357	24.644	59	16	107	91	0.325	-0.626
Test	13	28	60.393	28.510	62	15	125	110	0.381	-0.452
Overall	13	187	58.947	31.653	56	11	201	190	1.009	1.973
Training	14	52	71.962	42.654	64	12	192	180	0.655	-0.158
Validation	14	11	78.273	37.763	79	27	161	134	0.650	-0.424
Test	14	11	81.545	67.675	61	16	254	238	1.468	1.080
Overall	14	74	74.324	45.912	64	12	254	242	1.193	1.966
Training	15	80	66.713	36.058	57	11	209	198	1.015	1.539
Validation	15	17	75.353	32.270	68	27	144	117	0.610	-0.604
Test	15	17	77.824	38.439	75	13	156	143	0.530	-0.380
Overall	15	114	69.658	35.870	63	11	209	198	0.880	0.991
Training	16	104	53.221	33.078	49	12	183	171	1.510	2.768
Validation	16	22	59.091	35.355	52	16	156	140	1.138	0.705
Test	16	22	63.818	31.718	63	16	117	101	0.145	-1.247
Overall	16	148	55.669	33.240	50	12	183	171	1.263	1.770
Training	17	104	82.558	40.352	75	25	225	200	1.251	1.575
Validation	17	22	73.136	36.691	69	10	146	136	0.062	-1.037
Test	17	22	64.909	27.521	63	29	135	106	1.062	0.411
Overall	17	148	78.534	38.536	71	10	225	215	1.177	1.695
Training	18	106	69.274	35.429	61	15	223	208	1.424	2.736
Validation	18	23	86.739	42.256	74	22	198	176	0.795	0.170
Test	18	23	83.696	50.190	69	16	253	237	1.650	3.207
Overall	18	152	74.099	39.430	65	15	253	238	1.516	3.283
Training	19	85	74.824	38.528	73	15	191	176	0.796	0.144
Validation	19	18	74.611	40.873	59	23	141	118	0.468	-1.336
Test	19	19	63.684	28.642	67	20	134	114	0.605	-0.133
Overall	19	122	73.057	37.464	69	15	191	176	0.792	0.081
Training	20	83	65.759	33.645	64	12	166	154	0.507	-0.073
Validation	20	18	62.611	31.262	59	21	149	128	1.170	1.109

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Test	20	18	69.722	31.353	68	12	149	137	0.613	0.304
Overall	20	119	65.882	32.756	64	12	166	154	0.617	0.197
Training	21	62	61.016	36.630	54	11	177	166	1.229	1.301
Validation	21	14	57.786	38.455	43	15	138	123	1.040	-0.129
Test	21	13	54.385	29.287	54	18	136	118	1.397	1.950
Overall	21	89	59.539	35.657	54	11	177	166	1.271	1.346
Training	22	106	47.255	26.565	43	10	173	163	1.803	5.230
Validation	22	23	35.522	18.240	33	11	64	53	0.117	-1.603
Test	22	23	53.913	21.500	56	14	108	94	0.321	0.034
Overall	22	152	46.487	25.173	45	10	173	163	1.571	4.778
Training	23	25	59.520	31.329	53	15	116	101	0.318	-1.276
Validation	23	6	62.667	36.297	60	22	117	95	0.241	-1.715
Test	23	5	55.000	40.957	46	14	123	109	0.696	-1.272
Overall	23	36	59.417	32.520	54	14	123	109	0.432	-1.082
Training	24	41	64.585	29.552	66	13	122	109	0.027	-0.639
Validation	24	8	81.375	53.184	67	32	201	169	1.255	0.337
Test	24	9	43.778	19.873	49	11	68	57	-0.289	-1.573
Overall	24	58	63.672	33.507	62	11	201	190	1.123	3.089
Training	25	55	47.273	24.846	43	9	111	102	0.677	-0.301
Validation	25	12	53.583	30.714	56	10	104	94	0.048	-1.483
Test	25	12	47.917	19.783	50	19	88	69	0.446	-0.772
Overall	25	79	48.329	24.913	45	9	111	102	0.560	-0.480

APPENDIX F

KEY WORD COUNT DESCRIPTIVE STATISTICS

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Overall	Overall	4783	4.021	3.266	3	1	28	27	1.987	5.222
Training	Overall	3346	4.012	3.241	3	1	26	25	1.966	4.881
Validation	Overall	719	3.917	3.113	3	1	21	20	1.872	4.453
Test	Overall	718	4.169	3.519	3	1	28	27	2.096	6.492
Training	0	762	4.407	3.589	3	1	21	20	1.633	2.815
Validation	0	164	4.037	3.123	3	1	16	15	1.525	2.300
Test	0	163	4.939	4.420	3	1	28	27	1.950	5.046
Overall	0	1089	4.431	3.667	3	1	28	27	1.767	3.928
Training	1	338	5.473	4.382	4	1	26	25	1.444	2.366
Validation	1	72	5.653	4.296	5	1	21	20	1.389	1.679
Test	1	73	5.548	4.425	4	1	25	24	1.639	3.990
Overall	1	483	5.511	4.367	4	1	26	25	1.475	2.572
Training	2	142	3.711	2.800	3	1	15	14	1.943	4.348
Validation	2	30	4.367	3.399	3	1	15	14	1.351	1.474
Test	2	31	3.806	2.664	3	1	10	9	0.795	-0.494
Overall	2	203	3.823	2.870	3	1	15	14	1.727	3.343
Training	3	95	4.884	4.319	3	1	23	22	1.875	3.775
Validation	3	20	4.400	3.992	3	1	13	12	1.158	-0.019
Test	3	21	3.190	2.676	2	1	9	8	0.999	-0.573
Overall	3	136	4.551	4.082	3	1	23	22	1.860	3.902
Training	4	153	3.444	2.902	3	1	20	19	2.420	8.039
Validation	4	33	2.848	2.017	2	1	8	7	1.595	1.571
Test	4	33	3.000	1.887	2	1	7	6	0.946	-0.178
Overall	4	219	3.288	2.655	3	1	20	19	2.444	8.804
Training	5	127	4.039	3.118	3	1	14	13	1.368	1.241
Validation	5	28	3.286	2.537	3	1	10	9	1.358	1.157
Test	5	27	3.778	2.607	3	1	9	8	0.862	-0.584
Overall	5	182	3.885	2.963	3	1	14	13	1.372	1.343
Training	6	102	4.108	2.965	3	1	15	14	1.300	1.362
Validation	6	22	3.955	2.609	4	1	10	9	0.877	-0.423
Test	6	22	4.682	3.920	3	1	13	12	1.079	-0.275
Overall	6	146	4.171	3.063	3	1	15	14	1.302	1.160
Training	7	125	4.000	2.868	3	1	15	14	1.556	2.633
Validation	7	27	2.852	2.365	2	1	8	7	1.262	0.095
Test	7	27	3.852	2.699	3	1	11	10	1.225	0.608
Overall	7	179	3.804	2.789	3	1	15	14	1.514	2.402
Training	8	175	3.423	2.668	3	1	17	16	2.208	6.256
Validation	8	38	2.974	2.272	2	1	9	8	1.405	0.951
Test	8	37	3.730	2.775	3	1	12	11	1.228	0.710
Overall	8	250	3.400	2.627	3	1	17	16	1.998	5.026

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Training	9	49	2.776	1.723	2	1	7	6	0.820	-0.449
Validation	9	10	4.900	5.724	3	1	20	19	1.745	1.906
Test	9	11	1.909	1.136	2	1	4	3	0.901	-0.728
Overall	9	70	2.943	2.697	2	1	20	19	3.771	20.310
Training	10	73	3.699	2.228	3	1	11	10	1.371	1.853
Validation	10	16	3.250	1.238	3	2	5	3	0.346	-1.596
Test	10	15	4.333	3.288	3	1	13	12	1.300	0.692
Overall	10	104	3.721	2.292	3	1	13	12	1.625	2.998
Training	11	73	2.973	1.951	3	1	13	12	2.209	7.874
Validation	11	16	2.875	2.125	2	1	9	8	1.520	1.774
Test	11	15	2.533	1.060	3	1	4	3	-0.081	-1.362
Overall	11	104	2.894	1.869	3	1	13	12	2.167	7.607
Training	12	98	3.286	2.046	3	1	13	12	1.983	4.873
Validation	12	21	3.190	2.112	3	1	10	9	1.581	2.569
Test	12	21	3.476	2.768	3	1	11	10	1.803	2.347
Overall	12	140	3.300	2.161	3	1	13	12	1.990	4.463
Training	13	131	3.901	2.651	3	1	17	16	1.638	4.026
Validation	13	28	3.500	1.876	3	1	8	7	0.844	-0.352
Test	13	28	4.607	3.583	3	1	13	12	0.981	-0.162
Overall	13	187	3.947	2.717	3	1	17	16	1.561	3.164
Training	14	52	2.904	2.569	2	1	19	18	4.700	26.768
Validation	14	11	3.000	1.414	3	1	6	5	0.386	-0.364
Test	14	11	2.273	1.902	1	1	7	6	1.310	0.643
Overall	14	74	2.824	2.331	3	1	19	18	4.568	28.676
Training	15	80	3.238	2.517	3	1	17	16	2.702	10.372
Validation	15	17	3.176	2.651	2	1	10	9	0.995	0.016
Test	15	17	3.059	1.983	3	1	6	5	0.286	-1.613
Overall	15	114	3.202	2.447	3	1	17	16	2.287	8.333
Training	16	104	3.510	2.203	3	1	13	12	1.570	3.207
Validation	16	22	3.636	2.479	3	1	12	11	1.831	3.335
Test	16	22	3.727	2.640	3	1	11	10	1.171	0.458
Overall	16	148	3.561	2.298	3	1	13	12	1.592	2.898
Training	17	104	2.865	1.801	3	1	9	8	1.355	1.901
Validation	17	22	4.636	3.259	3	1	11	10	0.688	-1.040
Test	17	22	3.000	1.773	3	1	7	6	0.979	0.065
Overall	17	148	3.149	2.152	3	1	11	10	1.505	2.121
Training	18	106	3.330	2.258	3	1	11	10	1.424	1.727
Validation	18	23	3.000	1.931	3	1	8	7	1.378	1.237
Test	18	23	3.000	2.374	2	1	12	11	2.378	6.235
Overall	18	152	3.230	2.221	3	1	12	11	1.631	2.688
Training	19	85	3.494	2.223	3	1	13	12	1.668	3.225
Validation	19	18	4.056	3.489	3	1	14	13	1.337	1.157
Test	19	19	3.158	1.608	3	1	6	5	0.442	-1.251
Overall	19	122	3.525	2.364	3	1	14	13	1.772	3.967
Training	20	83	3.386	2.305	3	1	12	11	1.745	3.187
Validation	20	18	4.111	3.179	3	1	11	10	0.901	-0.532

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Test	20	18	4.167	3.618	3	1	14	13	1.386	0.778
Overall	20	119	3.613	2.675	3	1	14	13	1.672	2.503
Training	21	62	3.371	2.588	3	1	16	15	2.254	7.382
Validation	21	14	3.571	2.065	4	1	8	7	0.684	-0.513
Test	21	13	5.308	4.498	4	1	16	15	1.208	0.296
Overall	21	89	3.685	2.914	3	1	16	15	2.164	5.945
Training	22	106	2.830	1.920	2	1	9	8	1.312	0.956
Validation	22	23	4.652	4.030	3	1	19	18	1.922	4.256
Test	22	23	3.174	1.193	3	1	6	5	0.296	-0.546
Overall	22	152	3.158	2.356	2	1	19	18	2.607	12.149
Training	23	25	5.040	3.565	4	2	17	15	1.606	2.583
Validation	23	6	4.333	2.422	4	2	9	7	1.014	-0.544
Test	23	5	4.800	3.033	7	1	7	6	-0.322	-2.196
Overall	23	36	4.889	3.267	4	1	17	16	1.544	2.951
Training	24	41	7.341	4.453	6	1	17	16	0.439	-1.117
Validation	24	8	5.875	4.612	5	1	14	13	0.575	-1.348
Test	24	9	7.333	5.809	5	2	19	17	0.695	-0.923
Overall	24	58	7.138	4.639	6	1	19	18	0.575	-0.793
Training	25	55	4.945	4.253	2	1	15	14	0.910	-0.540
Validation	25	12	3.167	1.697	3	1	6	5	0.582	-1.271
Test	25	12	5.417	3.895	5	1	13	12	0.769	-0.842
Overall	25	79	4.747	3.943	3	1	15	14	1.064	-0.071

APPENDIX G

AGGREGATED DISTRACTOR WORD COUNT DESCRIPTIVE STATISTICS

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Overall	Overall	4783	15.584	11.744	11	3	104	101	2.123	6.135
Training	Overall	3346	15.522	11.816	11	3	104	101	2.185	6.482
Validation	Overall	719	15.348	11.030	12	3	99	96	2.141	7.158
Test	Overall	718	16.107	12.093	12	3	79	76	1.814	3.790
Training	0	762	16.980	13.077	13	3	104	101	1.880	4.787
Validation	0	164	15.976	11.957	11	3	68	65	1.540	2.235
Test	0	163	19.147	14.977	13	3	79	76	1.495	1.947
Overall	0	1089	17.153	13.238	12	3	104	101	1.790	4.026
Training	1	338	20.541	16.384	16	4	99	95	1.621	3.174
Validation	1	72	22.528	17.023	18	4	99	95	1.655	4.026
Test	1	73	21.247	16.137	17	4	64	60	1.170	0.438
Overall	1	483	20.944	16.425	16	4	99	95	1.571	2.990
Training	2	142	14.789	10.180	11	3	59	56	1.778	3.546
Validation	2	30	15.567	11.996	12	4	58	54	1.817	3.070
Test	2	31	16.000	12.315	13	3	48	45	0.900	-0.268
Overall	2	203	15.089	10.758	11	3	59	56	1.653	2.805
Training	3	95	17.389	13.700	13	4	79	75	2.252	6.372
Validation	3	20	17.050	11.358	12	6	43	37	0.970	-0.420
Test	3	21	13.952	10.166	9	4	35	31	0.717	-0.948
Overall	3	136	16.809	12.871	12	4	79	75	2.099	6.094
Training	4	153	13.007	9.992	10	4	75	71	3.060	13.312
Validation	4	33	10.242	7.475	8	4	37	33	2.054	3.854
Test	4	33	11.485	6.933	9	4	33	29	1.236	0.968
Overall	4	219	12.361	9.272	10	4	75	71	2.995	13.623
Training	5	127	15.189	11.840	11	4	63	59	1.793	3.327
Validation	5	28	13.357	7.558	12	4	36	32	1.183	1.286
Test	5	27	15.000	8.440	12	4	32	28	0.654	-0.960
Overall	5	182	14.879	10.806	11	4	63	59	1.797	3.802
Training	6	102	16.294	11.781	12	4	51	47	1.232	0.793
Validation	6	22	15.318	7.518	14	4	33	29	0.589	-0.683
Test	6	22	16.773	12.243	12	4	44	40	0.766	-0.709
Overall	6	146	16.219	11.258	13	4	51	47	1.186	0.782
Training	7	125	14.912	9.819	11	4	67	63	1.880	5.672
Validation	7	27	13.185	8.344	9	4	33	29	0.914	-0.505
Test	7	27	14.519	7.356	13	5	35	30	0.952	0.153
Overall	7	179	14.592	9.253	11	4	67	63	1.767	5.295
Training	8	175	14.017	10.324	10	4	69	65	2.181	6.027
Validation	8	38	12.026	8.388	9	4	34	30	1.217	0.377
Test	8	37	13.622	9.105	9	4	36	32	0.868	-0.608
Overall	8	250	13.656	9.870	10	4	69	65	1.990	5.291

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Training	9	49	10.918	10.138	8	3	63	60	3.204	12.414
Validation	9	10	17.900	19.762	11	4	69	65	1.695	1.565
Test	9	11	9.455	3.446	9	4	15	11	0.036	-1.233
Overall	9	70	11.686	11.444	9	3	69	66	3.336	12.444
Training	10	73	13.479	8.362	11	4	49	45	1.804	3.714
Validation	10	16	13.125	2.680	14	9	17	8	-0.144	-1.386
Test	10	15	17.800	15.275	12	4	63	59	1.775	2.457
Overall	10	104	14.048	9.168	12	4	63	59	2.502	8.423
Training	11	73	13.589	7.679	11	4	35	31	0.864	-0.098
Validation	11	16	12.875	6.109	11	4	25	21	0.588	-0.898
Test	11	15	11.267	4.480	10	4	17	13	-0.137	-1.359
Overall	11	104	13.144	7.074	11	4	35	31	0.925	0.281
Training	12	98	13.245	7.583	11	4	45	41	1.828	3.444
Validation	12	21	15.000	7.880	13	6	35	29	0.960	-0.057
Test	12	21	13.429	9.584	10	4	41	37	1.604	1.905
Overall	12	140	13.536	7.916	11	4	45	41	1.695	2.764
Training	13	131	15.626	9.498	13	4	58	54	1.594	3.353
Validation	13	28	14.107	7.146	12	7	39	32	1.645	2.883
Test	13	28	17.143	10.309	14	4	40	36	0.684	-0.604
Overall	13	187	15.626	9.305	13	4	58	54	1.483	2.738
Training	14	52	12.058	8.428	11	4	65	61	4.788	27.108
Validation	14	11	11.636	3.009	10	8	17	9	0.638	-1.259
Test	14	11	9.636	5.870	9	4	26	22	1.823	2.609
Overall	14	74	11.635	7.504	10	4	65	61	4.980	31.967
Training	15	80	10.950	6.242	9	4	41	37	1.954	5.693
Validation	15	17	11.412	6.462	9	4	30	26	1.399	1.562
Test	15	17	13.647	6.509	15	3	26	23	0.048	-1.259
Overall	15	114	11.421	6.329	10	3	41	38	1.557	3.555
Training	16	104	13.452	7.712	11	4	42	38	1.339	1.460
Validation	16	22	12.818	5.654	11	5	24	19	0.494	-1.063
Test	16	22	15.182	10.693	10	5	43	38	1.215	0.350
Overall	16	148	13.615	7.941	11	4	43	39	1.421	1.833
Training	17	104	12.481	5.139	12	4	40	36	1.601	6.514
Validation	17	22	15.000	7.703	12	8	40	32	1.853	3.025
Test	17	22	12.409	5.812	13	4	29	25	0.757	0.800
Overall	17	148	12.845	5.712	12	4	40	36	1.773	5.967
Training	18	106	13.075	6.966	11	4	38	34	1.381	1.954
Validation	18	23	10.435	5.333	9	5	24	19	1.368	0.698
Test	18	23	14.739	8.114	12	4	39	35	1.151	1.216
Overall	18	152	12.928	6.996	11	4	39	35	1.415	2.119
Training	19	85	12.165	6.892	10	5	39	34	1.756	3.064
Validation	19	18	14.944	7.158	14	6	34	28	0.964	0.372
Test	19	19	12.316	5.618	11	7	24	17	0.747	-0.952
Overall	19	122	12.598	6.770	10	5	39	34	1.543	2.335
Training	20	83	13.530	8.956	11	4	59	55	2.573	8.730
Validation	20	18	19.556	11.278	18	4	42	38	0.390	-1.060

Set	Domain	N	Mean	SD	Median	Minimum	Maximum	Range	Skew	Kurtosis
Test	20	18	12.111	5.738	11	5	30	25	1.546	2.594
Overall	20	119	14.227	9.174	11	4	59	55	2.052	5.420
Training	21	62	13.161	7.795	11	4	51	47	2.155	7.065
Validation	21	14	12.357	5.063	12	4	20	16	-0.009	-1.274
Test	21	13	16.231	11.159	11	8	45	37	1.457	0.878
Overall	21	89	13.483	8.017	11	4	51	47	2.125	6.200
Training	22	106	11.613	7.900	9	4	46	42	2.541	7.072
Validation	22	23	16.957	13.976	11	5	55	50	1.590	1.713
Test	22	23	10.522	4.926	10	5	22	17	0.817	-0.364
Overall	22	152	12.257	8.916	9	4	55	51	2.620	7.883
Training	23	25	20.640	15.094	16	9	61	52	1.502	1.354
Validation	23	6	16.000	5.797	15	8	23	15	-0.005	-1.825
Test	23	5	14.600	7.503	14	5	26	21	0.281	-1.398
Overall	23	36	19.028	13.179	15	5	61	56	1.834	3.104
Training	24	41	28.000	17.407	27	5	76	71	0.768	0.078
Validation	24	8	25.250	14.069	27	8	42	34	-0.052	-2.060
Test	24	9	27.889	20.727	20	5	69	64	0.629	-0.896
Overall	24	58	27.603	17.267	27	5	76	71	0.746	0.058
Training	25	55	19.927	17.430	10	4	62	58	0.938	-0.503
Validation	25	12	13.500	6.842	13	4	24	20	0.243	-1.375
Test	25	12	20.000	14.000	14	5	47	42	0.599	-1.189
Overall	25	79	18.962	15.811	11	4	62	58	1.082	0.013

APPENDIX H

FIVE OPTION ITEM COUNT DESCRIPTIVE STATISTICS

Set	Domain	N	Count
Overall	Overall	4783	4415
Training	Overall	3346	3084
Validation	Overall	719	668
Test	Overall	718	663
Training	0	762	693
Validation	0	164	153
Test	0	163	149
Overall	0	1089	995
Training	1	338	321
Validation	1	72	69
Test	1	73	71
Overall	1	483	461
Training	2	142	130
Validation	2	30	28
Test	2	31	29
Overall	2	203	187
Training	3	95	82
Validation	3	20	17
Test	3	21	20
Overall	3	136	119
Training	4	153	146
Validation	4	33	33
Test	4	33	31
Overall	4	219	210
Training	5	127	116
Validation	5	28	27
Test	5	27	25
Overall	5	182	168
Training	6	102	92
Validation	6	22	21
Test	6	22	21
Overall	6	146	134
Training	7	125	118

Set	Domain	N	Count
Validation	7	27	25
Test	7	27	25
Overall	7	179	168
Training	8	175	165
Validation	8	38	36
Test	8	37	35
Overall	8	250	236
Training	9	49	45
Validation	9	10	10
Test	9	11	10
Overall	9	70	65
Training	10	73	65
Validation	10	16	14
Test	10	15	15
Overall	10	104	94
Training	11	73	65
Validation	11	16	12
Test	11	15	14
Overall	11	104	91
Training	12	98	92
Validation	12	21	21
Test	12	21	19
Overall	12	140	132
Training	13	131	125
Validation	13	28	27
Test	13	28	25
Overall	13	187	177
Training	14	52	44
Validation	14	11	10
Test	14	11	9
Overall	14	74	63
Training	15	80	64
Validation	15	17	15
Test	15	17	14
Overall	15	114	93
Training	16	104	100
Validation	16	22	21
Test	16	22	19

Set	Domain	N	Count
Overall	16	148	140
Training	17	104	98
Validation	17	22	20
Test	17	22	22
Overall	17	148	140
Training	18	106	100
Validation	18	23	19
Test	18	23	20
Overall	18	152	139
Training	19	85	79
Validation	19	18	17
Test	19	19	17
Overall	19	122	113
Training	20	83	77
Validation	20	18	17
Test	20	18	15
Overall	20	119	109
Training	21	62	59
Validation	21	14	12
Test	21	13	13
Overall	21	89	84
Training	22	106	102
Validation	22	23	21
Test	22	23	22
Overall	22	152	145
Training	23	25	22
Validation	23	6	4
Test	23	5	4
Overall	23	36	30
Training	24	41	33
Validation	24	8	7
Test	24	9	7
Overall	24	58	47
Training	25	55	51
Validation	25	12	12
Test	25	12	12
Overall	25	79	75

APPENDIX I

AVERAGE COSINE SIMILARITY MATRIX FOR DOMAINS 1-25

Domain	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	.426																								
2	.425	.495																							
3	.422	.471	.466																						
4	.396	.457	.434	.446																					
5	.403	.441	.440	.416	.446																				
6	.416	.463	.446	.436	.429	.456																			
7	.401	.470	.446	.432	.419	.440	.465																		
8	.397	.469	.457	.445	.417	.434	.450	.521																	
9	.386	.470	.451	.442	.414	.429	.452	.507	.519																
10	.414	.474	.452	.457	.425	.449	.445	.472	.474	.486															
11	.408	.490	.470	.468	.442	.460	.465	.503	.499	.489	.534														
12	.411	.488	.466	.458	.437	.455	.463	.465	.481	.477	.497	.505													
13	.399	.456	.438	.416	.410	.428	.455	.431	.441	.430	.442	.453	.473												
14	.389	.481	.456	.451	.425	.443	.466	.494	.506	.477	.505	.492	.457	.524											
15	.395	.474	.449	.442	.435	.443	.462	.457	.470	.458	.483	.477	.454	.489	.484										
16	.400	.481	.459	.442	.424	.445	.477	.484	.489	.460	.486	.478	.470	.493	.478	.509									
17	.393	.482	.452	.441	.422	.443	.471	.463	.473	.458	.479	.481	.457	.497	.478	.487	.508								
18	.401	.477	.454	.439	.428	.446	.467	.472	.471	.455	.485	.470	.451	.484	.472	.485	.481	.481							
19	.407	.488	.464	.452	.428	.451	.472	.481	.487	.469	.491	.488	.464	.494	.479	.497	.486	.482	.506						
20	.410	.482	.458	.455	.426	.456	.461	.476	.477	.478	.495	.482	.444	.489	.470	.479	.476	.473	.484	.490					
21	.408	.493	.475	.450	.436	.454	.479	.481	.488	.466	.492	.493	.471	.497	.483	.504	.495	.486	.503	.486	.517				
22	.389	.469	.454	.445	.421	.436	.454	.493	.493	.465	.505	.474	.444	.497	.472	.489	.466	.474	.484	.481	.488	.518			
23	.419	.449	.447	.410	.432	.430	.435	.423	.424	.424	.430	.441	.434	.429	.433	.449	.436	.437	.443	.431	.452	.424	.449		
24	.444	.454	.465	.416	.445	.443	.430	.427	.415	.432	.432	.443	.429	.418	.420	.437	.427	.430	.438	.432	.450	.416	.469	.509	
25	.399	.410	.410	.381	.398	.402	.396	.386	.374	.395	.396	.397	.393	.384	.390	.396	.391	.394	.395	.399	.405	.382	.408	.434	.396