Combining Independent Modules to Solve Multiple-choice Synonym and Analogy Problems

Jeffrey Bigham
Dept. of Computer Science
Princeton University
Princeton, NJ 08544
jbigham@princeton.edu

Michael L. Littman
Dept. of Computer Science
Rutgers University
Piscataway, NJ 08854-8019
mlittman@cs.rutgers.edu

Victor Shnayder
Dept. of Computer Science
Princeton University
Princeton, NJ 08544
shnayder@cs.princeton.edu

Peter D. Turney
Inst. of Information Tech.
National Research Council
Ont., Canada, K1A 0R6
peter.turney@nrc.ca

Abstract

Existing statistical approaches to natural language problems are very coarse approximations to the true complexity of language processing. As such, no single technique will be best for all problem instances. Many researchers are examining ensemble methods that combine the output of successful, separately developed modules to create more accurate solutions. This paper examines three merging rules for combining probability distributions: the well known mixture rule, the logarithmic rule, and a novel product rule. These rules were applied with state-of-the-art results to two problems commonly used to assess human mastery of lexical semantics—synonym questions and analogy questions.

1 INTRODUCTION

Asked to articulate the relationship between the words broad and road, you might consider a number of possibilities. Orthographically, the second can be derived from the first by deleting the initial letter, while semantically, the first can modify the second to indicate above average width. Many possible relationships would need to be considered, depending on the context. In addition, many different computational approaches could be brought to bear, leaving a designer of a natural language processing system with some difficult choices. A sound software engineering approach is to develop separate modules using independent strategies, then to combine the output of the modules to produce a unified solver.

The concrete problem treated here is predicting the correct answers to multiple-choice questions. Each instance consists of a context and a finite set of choices, one of which is correct. Modules produce a probability distribution over the choices and a merging rule is used to combine these distributions into one. This distribution, along with relevant utilities, can then be used to select a candidate answer from the set of choices. The merging rules we considered are parameterized, and we set parameters by a maximum likelihood approach on a collection of training instances.

Many problems can be cast in this multiple-choice framework, including optical digit recognition and text categorization. This paper looks at a more direct example, that of multiple-choice questions on standardized exams, specifically synonym and verbal analogy problems. The paper offers two main contributions: a collection of novel modules for addressing problems in lexical semantics, including evaluations of these modules on challenging problems; and a novel product rule for combining module outputs, including a comparison to other similar merging rules.

Section 2 formalizes the problem addressed in this paper and introduces the three merging rules we study in detail: the mixture rule, the logarithmic rule, and the product rule. Section 3 presents empirical results on artificial problems, synonym problems, and analogy problems. Section 4 summarizes and wraps up.

2 MODULE COMBINATION

The following synonym question is a typical multiple-choice question: hidden:: (a) laughable, (b) veiled, (c) ancient, (d) revealed. The stem, hidden, is the question. There are $k = 4$ choices, and the question writer asserts that exactly one (in this case, (b)) has the same meaning as the stem word. The accuracy of a solver is measured by its fraction of correct answers on a set of ℓ testing instances.

In our setup, knowledge about the multiple-choice task is encapsulated in a set of n modules, each of which can take a question instance and return a probability distribution over the k choices. For a synonym task, one module might be a statistical approach that makes

judgments based on analyses of word co-occurrence, while another might use a thesaurus to identify promising candidates. These modules are applied to a training set of m instances, producing probabilistic "forecasts"; $p_{ij}^h \geq 0$ represents the probability assigned by module $1 \leq i \leq n$ to choice $1 \leq j \leq k$ on training instance $1 \leq h \leq m$. The estimated probabilities are distributions of the choices for each module i on each instance h: $\sum_j p_{ij}^h = 1$.

2.1 MERGING RULES

The merging rules we considered are parameterized by a set of weights w_i, one for each module. For a given merging rule, a setting of the weight vector w induces a probability distribution over the choices for any instance. Let $D_j^{h,w}$ be the probability assigned by the merging rule to choice j of training instance h when the weights are set to w. Let $1 \leq a(h) \leq k$ be the correct answer for instance h. We set weights to maximize the likelihood of the training data: $w = \text{argmax}_{w'} \prod_h D_{a(h)}^{h,w'}$. The same weights maximize the *mean likelihood*, the geometric mean of the probabilities assigned to correct answers, $\prod_h \sqrt[m]{D_{a(h)}^{h,w}}$.

We focus on three merging rules in this paper. The *mixture rule* combines module outputs using a weighted sum and can be written $M_j^{h,w} = \sum_i w_i p_{ij}^h$, where $D_j^{h,w} = M_j^{h,w} / (\sum_j M_j^{h,w})$ is the probability assigned to choice j of instance h and $0 \leq w_i \leq 1$. The rule can be justified by assuming each instance's answer is generated by a single module chosen via the distribution $w_i / \sum_i w_i$.

The *logarithmic rule* combines the logarithm of module outputs by $L_j^{h,w} = \exp(\sum_i w_i \ln p_{ij}^h) = \prod_i (p_{ij}^h)^{w_i}$, where $D_j^{h,w} = L_j^{h,w} / (\sum_j L_j^{h,w})$ is the probability the rule assigns to choice j of instance h. The weight w_i indicates how to scale the module probabilities before they are combined multiplicatively. Note that modules that output zero probabilities must be modified before this rule can be used.

The *product rule* is $P_j^{h,w} = \prod_i (w_i p_{ij}^h + (1 - w_i)/k)$, where $D_j^{h,w} = P_j^{h,w} / (\sum_j P_j^{h,w})$ is the probability the rule assigns to choice j. The weight $0 \leq w_i \leq 1$ indicates how module i's output should be mixed with a uniform distribution (or a prior, more generally) before outputs are combined multiplicatively. As with the mixture and logarithmic rules, a module with a weight of zero has no influence on the final assignment of probabilities. Note that the product and logarithmic rules coincide when weights are all zeroes and ones, but differ in how distributions are scaled for intermediate weights. We have not yet found this difference to be significant empirically.

2.2 DERIVATION OF PRODUCT RULE

In this section, we provide a justification for combining distributions multiplicatively, as in both the product and logarithmic rules. Our analysis assumes modules are *calibrated* and *independent*. The output of a calibrated module can be treated as a valid probability distribution—of all the times the module outputs 0.8 for a choice, 80% of these should be correct. Note that a uniform distribution—the output of any module when its weight is zero for both rules—is guaranteed to be calibrated because the output is always $1/k$ and $1/k$ of these will be correct. Modules are independent if their outputs are independent given the correct answer. We next argue that our parameterization of the product rule can compensate for a lack of calibration and independence.

Use of Weights. First, module weights can improve the calibration of the module outputs. Consider a module i that assigns a probability of 1 to its best guess and 0 to the other three choices. If the module is correct 85% of the time, setting $w_i = 0.85$ in the product rule results in adjusting the output of the module to be 85% for its best guess and 5% for each of the lesser choices. This output is properly calibrated and also maximizes the likelihood of the data.[1]

Second, consider the situation of two modules with identical outputs. Unless they are perfectly accurate, such modules are not independent and combining their outputs multiplicatively results in "double counting" the evidence. However, assigning either module a weight of zero renders the modules independent. Once again, such a setting of the weights maximizes the likelihood of the data.

Multiplicative Combination. We now argue that independent, calibrated modules should be combined multiplicatively. Let A^h be the random variable representing the correct answer to instance h. Let $\hat{p}_i^h = \langle p_{i1}^h, \ldots, p_{ik}^h \rangle$ be the output vector of module i on instance h. We would like to compute the probability the correct answer is j given the module outputs, $\Pr(A^h = j | \hat{p}_1^h, \ldots, \hat{p}_n^h)$, which we can rewrite with Bayes rule as

$$\Pr(\hat{p}_1^h, \ldots, \hat{p}_n^h | A^h = j) \Pr(A^h = j) / \Pr(\hat{p}_1^h, \ldots, \hat{p}_n^h). \quad (1)$$

Assuming independence, and using Z as a normalization factor, Expression 1 can be decomposed into $\Pr(\hat{p}_1^h | A^h = j) \cdots \Pr(\hat{p}_n^h | A^h = j) \Pr(A^h = j) / Z$. Ap-

[1]The logarithmic rule can also calibrate this module, as long as its output is renormalized after adding a small constant, say, $\varepsilon = 0.00001$, to avoid logarithms of $-\infty$. In this case, $w_i \approx .2461$ works, although the appropriate weight varies with ε.

plying Bayes rule to the individual factors, we get

$$\Pr(A^h = j|\hat{p}_1^h) \cdots \Pr(A^h = j|\hat{p}_n^h) / \Pr(A^h = j)^{n-1} / Z' \quad (2)$$

by collecting constant factors into the normalization factor Z'. Using the calibration assumption, Expression 2 simplifies to $\prod_i p_{ij}^h / \Pr(A^h = j)^{n-1} / Z'$. Finally, we precisely recover the unweighted product rule using a final assumption of uniform priors on the choices, $\Pr(A^h = j) = 1/k$, which is a natural assumption for standardized tests.

2.3 WEIGHT OPTIMIZATION

For the experiments reported here, we adopted a straightforward approach to finding the weight vector w that maximizes the likelihood of the data. The weight optimizer reads in the output of the modules.[2] Let e_i be an n-vector of zeroes with one in the ith position, and let ϵ be small (0.01, in our experiments). An episode of optimization begins by setting the weight vector w to random values. We estimate the partial derivative of the log likelihood $S(w) = \sum_{h=1}^{m} \ln(D_{a(h)}^{h,w})$, by

$$\left.\frac{\partial S(w)}{\partial w_i}\right|_w \approx \frac{S(w + \epsilon e_i) - S(w - \epsilon e_i)}{2\epsilon}.$$

Let $\mathcal{D} = (\frac{\partial S(w)}{\partial w_1}, \ldots, \frac{\partial S(w)}{\partial w_n})$ be the vector of estimated partial derivatives. We modify $\mathcal{D}$ by zeroing out components that would result in updates violating constraints on the weights (mixture and product rules) and truncating values at 0.05 to avoid numeric instability. We then compute a step size $c = \min_i T/\mathcal{D}_i$, for threshold T (0.05, in our experiments), and take a step in weight space: $w := w + c\mathcal{D}$. Steps are taken until $||\mathcal{D}|| < \theta$ for some threshold θ (2, in our experiments) or NUM_STEPS are reached (500, in our experiments). The entire optimization procedure is repeated R times (10, in our experiments) from a new random starting point to minimize the influence of local minima.

2.4 RELATED WORK

Merging rules of various sorts have been studied for many years, and have gained prominence recently for natural language applications.

Use of the mixture rule and its variations is quite common. Recent examples include the work of Brill and Wu (1998) on part-of-speech tagging, Littman et al. (2002) on crossword-puzzle clues and Florian and Yarowsky (2002) on a word-sense disambiguation task. In all of these cases, the authors found that the merged output was a significant improvement on that of the powerful independently engineered component modules. We use the name "mixture rule" by analogy to the mixture of experts model (Jacobs et al. 1991), which combined expert opinions in an analogous way. In the forecasting literature, this rule is also known as the linear opinion pool; Jacobs (1995) provides a summary of the theory and applications of the mixture rule in this setting.

The logarithmic opinion pool of Heskes (1998) is the basis for our logarithmic rule. The paper argued that its form can be justified as an optimal way to minimize Kullback-Leibler divergence between the output of an ensemble of adaptive experts and target outputs. Boosting (Schapire 1999) also uses a logistic-regression-like rule to combine outputs of simple modules to perform state-of-the-art classification. The product of experts approach also combines distributions multiplicatively, and Hinton (1999) argues that this is an improvement over the "vaguer" probability judgments commonly resulting from the mixture rule. A survey by Xu et al. (1992) includes the equal-weights version of the mixture rule and a derivation of the unweighted product rule.

An important contribution of the current work is the product rule, which shares the simplicity of the mixture rule and the probabilistic justification of the logarithmic rule. We have not seen an analog of this rule in the forecasting or learning literatures.

3 EXPERIMENTAL RESULTS

We compared the three merging rules for artificial problems, synonym problems, and analogy problems.

3.1 ARTIFICIAL PROBLEMS

We present results on a set of artificial problems intended to illuminate the behavior of the merging rules. For each problem, we created a 5000-item training set and a 5000-item testing set. Each question was assigned an answer uniformly at random from five possible choices. We simulated a set of three modules (X, Y, and Z), with output generated in different ways. We then used the training data to set module weights for the mixture, logarithmic and product rules and report how they generalized to the testing set. Table 1 reports the results of applying each merging rule to each of the scenarios described next.

All Independent. In this variation, module X re-

[2]For the reasons suggested in the previous footnote, for each question and module, the optimizer adds 0.00001 to each output and renormalizes the distribution (scales it to add to one). We found that this is necessary for both the logarithmic and mixture rules, but not the product rule. Parameters were set by informal experimentation, but the results did not seem to be sensitive to their exact values.

Artificial Modules	All Independent	One Dependent	Two Dependent	Partly Dependent	Opt Out	4321
accuracy X (expected)	70.0%	70.0%	70.0%	70.0%	70.0%	40.0%
accuracy Y (expected)	70.0%	70.0%	70.0%	70.0%	70.0%	40.0%
accuracy Z (expected)	75.0%	75.0%	75.0%	75.0%	75.0%	40.0%
mixture	.6624	.3171	.0588	.1988	.5591	.2934
logarithmic	.6624	.5383	.3999	.4621	.6420	.3360
product	.6624	.5385	.3998	.4623	.6420	.3360
accuracy (all)	85.5%	75.7%	74.7%	75.8%	87.7%	51.2%

Table 1: Mean likelihood of correct answers for each merging rule on artificial problems.

ports the correct answer with probability 0.7 and otherwise chooses an answer uniformly at random from the remaining (incorrect) choices. Module Y independently reports the correct answer also with probability 0.7, and Module Z independently reports the correct answer with probability 0.75. We expect the product rule's weights to reflect the module accuracies, since the modules' outputs are independent given the correct answer. Hence, they should simply calibrate the module output distributions.

One Dependent. Module Z chooses correctly with probability 0.75. Module X independently chooses correctly with probability 0.7. Module Y chooses randomly with probability 0.0476, mimics X with probability 0.4762 and mimics Z with probability 0.4762, and therefore adds no information. We'd expect the product rule to assign X and Z weights of 0.7 and 0.75, respectively, and for Y to get no weight.

Two Dependent. Module Z reports the correct answer with probability 0.75. Modules X and Y independently mimic Z's output with probability 0.9091 and otherwise choose randomly. Since X and Y add no information, we'd expect their weights to go to zero.

Partly Dependent. Module Z reports the correct answer with probability 0.75. Module X mimics Z half the time and otherwise chooses correctly with probability 0.65. Thus, Module X adds some amount of information over that provided by Z. Similarly, Module Y mimics Z with probability 0.4, and X with probability 0.2, and otherwise chooses correctly with probability 0.65. This means Y adds little information.

In each of the examples, correctly merged probabilities can be computed by probabilistic inference in a simple belief network consisting of a hidden node representing the true answer and evidence nodes for each of the modules. In the first three cases, reasoning is relatively simple as the network structure is singly connected, and the product rule can be used to produce exact answers. In the Partly Dependent case, however, the network structure is multiply connected and the product rule produces an approximation of the true merged probabilities.

Opt Out. With probability 0.6967, Module Z answers correctly, with probability 0.0366 incorrectly, and with probability 0.2667 it "knows it doesn't know" and returns the uniform distribution. Counting the uniform distribution as an accuracy of 1 in 5, Z's accuracy is 0.75. However, Z's accuracy is higher when it has an opinion. Module X returns the correct answer with probability 0.7, and otherwise chooses randomly from the incorrect options. Module Y mimics X. The product rule is expected to assign weights of 0.95 for Module Z (its accuracy when it returns an answer) and 0.7 for X or Y with the other getting a weight of zero.

4321. This example has only four choices. All three modules work independently and use the same rule for assigning probabilities. Each module assigns the four choices the probabilities 0.4, 0.3, 0.2, and 0.1. For each of these values p, with probability p, the correct answer is assigned value p (while the others are assigned randomly). We'd expect the product rule to assign weights of 1.0 to all three modules since all are independent and calibrated.

Discussion. Table 1 presents the test set accuracy and mean likelihood (geometric mean of probabilities assigned to correct answers) for the six artificial problems. Results for all three merging rules are given. In all cases, the learned weights (not shown) for the product rule were close to their expected values (generally a bit low). They were also correctly set to zero for modules that added no independent information. In the three cases with significant module independence, the accuracy of the merged rule dominated that of the best single module by more than 10%, showing the potential benefits of combining distributions.

In all our examples, all three merging rules have the same accuracy. However, the mean likelihood shows that the product and logarithmic rules assign consistently higher probabilities to correct answers than the mixture rule. Multiplying distributions is a better way to combine evidence in these examples.

3.2 SYNONYMS

We constructed a training set of 431 4-choice synonym questions[3] and randomly divided them into 331 training questions and 100 testing questions. We created four modules, described next, and ran each module on the training set. We used the results to set the weights for the mixture, logarithmic, and product rules and evaluated the resulting synonym solver on the test set.

Module outputs, where applicable, were normalized to form a probability distribution by scaling them to add to one before merging.

LSA. Following Landauer and Dumais (1997), we used latent semantic analysis to recognize synonyms. Our LSA module queried the web interface developed at the University of Colorado (`http://lsa.colorado.edu/cgi-bin/LSA-one2many.html`), which has a 300-dimensional vector representation for each of tens of thousands of words. The similarity of two words is measured by the cosine of the angle between their corresponding vectors.

PMI-IR. Our PMI-IR (Pointwise Mutual Information - Information Retrieval) module used the AltaVista search engine to determine the number of web pages that contain the choice and stem in close proximity. We used the third scoring method (near each other, but not near `not`) designed by Turney (2001), since it performed best in this earlier study.

Thesaurus. Our Thesaurus module also used the web to measure stem–choice similarity. The module queried the Wordsmyth thesaurus online at `www.wordsmyth.net` and collected any words listed in the "Similar Words", "Synonyms", "Crossref. Syn.", and "Related Words" fields. The module created synonym lists for the stem and for each choice, then scored them according to their overlap.

Connector. Our Connector module used summary pages from querying Google (`google.com`) with pairs of words to estimate stem–choice similarity (20 summaries for each query). It assigned a score to a pair of words by taking a weighted sum of both the number of times they appear separated by one of the symbols [, ", :, ,, =, /, \, (,], means, defined, equals, synonym, whitespace, and and and the number of times dictionary or thesaurus appear anywhere in the Google summaries.

[3]Our synonym question set consisted of 80 TOEFL questions provided by ETS via Thomas Landauer, 50 ESL questions created by Donna Tatsuki for Japanese ESL students, 100 Reader's Digest Word Power questions gathered by Peter Turney, Mario Jarmasz, and Tad Stach, and 201 synonym pairs and distractors drawn from different sources including crossword puzzles by Jeffrey Bigham.

Synonym Solvers	Accuracy	Mean likelihood
LSA only	43.8%	.2669
PMI-IR only	69.0%	.2561
Thesaurus only	69.6%	.5399
Connector only	64.2%	.3757
All: mixture	80.2%	.5439
All: logarithmic	82.0%	.5977
All: product	80.0%	.5889

Table 2: Comparison of results for merging rules on synonym problems.

Reference	Accuracy	95% conf. interval
L & D (1997)	64.40%	53.75– 75.00%
non-native speakers	64.50%	53.75– 75.00%
Turney (2001)	73.75%	63.75– 82.25%
J & S (2002)	78.75%	70.00– 87.50%
T & C (2003)	81.25%	72.50– 90.00%
product rule	97.50%	93.75–100.00%

Table 3: Published TOEFL synonym results. Confidence intervals computed from 100k random samples.

Results. Table 2 presents the result of training and testing each of the four modules on synonym problems. The first four lines list the accuracy and mean likelihood obtained using each module individually. The highest accuracy is that of the Thesaurus module at 69.6%. All three merging rules were able to leverage the combination of the modules to improve performance to roughly 80%—statistically significantly better than the best individual module. It seems this domain lends itself very well to an ensemble approach.

Once again, although the accuracies of the merging rules are nearly identical, the product and logarithmic rules assign higher probabilities to correct answers, as evidenced by the mean likelihood. To illustrate the decision-theoretic implications of this difference, imagine the probability judgments were used in a system that receives a score of +1 for each right answer and $-1/2$ for each wrong answer, but can skip questions.[4] In this case, the system should make a guess whenever the highest probability choice is above 1/3. For the test questions, this translates to scores of 71.0 and 73.0 for the product and logarithmic rules, but only 57.5 for the mixture rule; it skips many more questions because it is insufficiently certain.

Related Work and Discussion. Landauer and Dumais (1997) introduced the Test of English as a Foreign Language (TOEFL) synonym task as a way of assess-

[4]The penalty value of $-1/2$ was chosen to illustrate this point. Standardized tests often use a penalty of $-1/(k-1)$, which grants random guessing and skipping equal utility.

ing the accuracy of a learned representation of lexical semantics. Several studies have since used the same data set for direct comparability; Table 3 presents these results.

The accuracy of LSA (Landauer and Dumais 1997) is statistically indistinguishable from that of a population of non-native English speakers on the same questions. PMI-IR (Turney 2001) performed better, but the difference is not statistically significant. Jarmasz and Szpakowicz (2002) give results for a number of relatively sophisticated thesaurus-based methods that looked at path length between words in the heading classifications of Roget's Thesaurus. Their best scoring method was a statistically significant improvement over the LSA results, but not over those of PMI-IR. Terra and Clarke (2003) studied a variety of corpus-based similarity metrics and measures of context and achieved a statistical tie with PMI-IR and the results from Roget's Thesaurus.

To compare directly to these results, we removed the 80 TOEFL instances from our collection and used the other 351 instances for training the product rule. Unlike the previous studies, we used training data to set the parameters of our method instead of selecting the best scoring method post hoc. The resulting accuracy was statistically significantly better than all previously published results, even though the individual modules performed nearly identically to their published counterparts. In addition, the fact that the upper confidence interval reaches 100% essentially means that the TOEFL test set is a "solved" problem—future studies along these lines will need to use a more challenging set of questions to show an improvement over our results.

3.3 ANALOGIES

Synonym questions are unique because of the existence of thesauri—reference books designed precisely to answer queries of this form. The relationships exemplified in analogy questions are quite a bit more varied and are not systematically compiled. For example, the analogy question cat:meow:: (a) mouse:scamper, (b) bird:peck, (c) dog:bark, (d) horse:groom, (e) lion:scratch requires that the reader recognize that (c) is the answer because both (c) and the stem are examples of the relation "X is the name of the sound made by Y". This type of common sense knowledge is rarely explicitly documented.

In addition to the computational challenge they present, analogical reasoning is recognized as an important component in cognition, including language comprehension (Lakoff and Johnson 1980) and high level perception (Chalmers et al. 1992). French (2002) surveys computational approaches to analogy making.

To study module merging for analogy problems, we collected 374 5-choice instances.[5] We randomly split the collection into 274 training instances and 100 testing instances.

We next describe the novel modules we developed for attacking analogy problems and present their results.

Phrase Vectors. We wish to score candidate analogies of the form A:B::C:D (A is to B as C is to D). The quality of a candidate analogy depends on the similarity of the relation R_1 between A and B to the relation R_2 between C and D. The relations R_1 and R_2 are not given to us; the task is to infer these relations automatically. One approach to this task is to create vectors r_1 and r_2 that represent features of R_1 and R_2, and then measure the similarity of R_1 and R_2 by the cosine of the angle between the vectors: $r_1 \cdot r_2 / \sqrt{(r_1 \cdot r_1)(r_2 \cdot r_2)}$.

We create a vector, r, to characterize the relationship between two words, X and Y, by counting the frequencies of 128 different short phrases containing X and Y. Phrases include "X for Y", "Y with X", "X in the Y", and "Y on X". We use these phrases as queries to AltaVista and record the number of hits (matching documents) for each query. This process yields a vector of 128 numbers for a pair of words X and Y. In experiments with our development set, we found that accuracy of this approach to scoring analogies improves when we use the logarithm of the frequency. The resulting vector r is a kind of *signature* of the relationship between X and Y.

For example, consider the analogy traffic:street:: water:riverbed. The words traffic and street tend to appear together in phrases such as "traffic in the street" and "street with traffic", but not in phrases such as "street on traffic" or "traffic for street. Similarly, water and riverbed may appear together as "water in the riverbed", but "riverbed on water" would be uncommon. Therefore, the cosine of the angle between the 128-vector r_1 for traffic:street and the 128-vector r_2 for water:riverbed would likely be relatively large.

Thesaurus Paths. Another way to characterize the semantic relationship, R, between two words, X and Y, is to find a path through a thesaurus or dictionary that connects X to Y or Y to X.

In our experiments, we use the WordNet thesaurus (Fellbaum 1998). We view WordNet as a di-

[5] Our analogy question set was constructed by the authors from books and web sites intended for students preparing for the Scholastic Aptitude Test (SAT), including 90 questions from unofficial SAT-prep websites, 14 questions ETS's web site, 190 questions scanned in from a book with actual SAT exams, and 80 questions typed from SAT guidebooks.

rected graph and perform a breadth-first search for paths from X to Y or Y to X. The directed graph has six kinds of links, *hypernym*, *hyponym*, *synonym*, *antonym*, *stem*, and *gloss*. For a given pair of words, X and Y, we consider all shortest paths in either direction up to three links. We score the candidate analogy by the maximum degree of similarity between any path for A and B and any path for C and D. The degree of similarity between paths is measured by their number of shared features: types of links, direction of the links, and shared words.

For example, consider the analogy evaporate :vapor::petrify:stone. The most similar pair of paths is:

evaporate → (gloss: *change into a vapor*) vapor and
petrify → (gloss: *change into stone*) stone.

These paths go in the same direction (from first to second word), they have the same type of links (gloss links), and they share words (*change* and *into*).

Lexical Relation Modules. We implemented a set of more specific modules using the WordNet thesaurus. Each module checks if the stem words match a particular relationship in the database. If they do not, the module returns the uniform distribution. Otherwise, it checks each choice pair and eliminates those that do not match. The relations tested are: **Synonym**, **Antonym**, **Hypernym**, **Hyponym**, **Meronym:substance**, **Meronym:part**, **Meronym:member**, **Holonym:substance**, and **Holonym:member**. These modules use some heuristics including a simple kind of lemmatization and synonym expansion to make matching more robust.

Similarity. Dictionaries are a natural source to use for solving analogies because definitions can express many possible relationships and are likely to make the relationships more explicit than they would be in general text. We implemented two definition similarity modules: **Similarity:dict** uses `Dictionary.com` for definitions and **Similarity:wordsmyth** uses `Wordsmyth.net`. Each module treats a word as a vector formed from the words in its definition. Given a potential analogy A:B::C:D, the module computes a vector similarity of the first words (A and C) and adds it to the vector similarity of the second words (B and D).

Results. We ran the 13 modules described above on our set of training and testing analogy instances, with the results appearing in Table 4. For the most part, individual module accuracy is near chance level (20%), although this is misleading because most of these modules only return answers for a small subset of instances. Some modules did not answer a single question on the test set. The most accurate individual module was the search-engine-based Phrase Vectors (PV) module. The results of merging all modules was only a slight improvement over PV alone, so we examined the effect of retraining without the PV module. The product rule resulted in a large improvement (though not statistically significant) over the best remaining individual module (37.0% vs. 29.4% for Similarity:wordsmyth).

Analogy Solvers	Accuracy	Mean likelihood
Phrase Vectors	38.2%	.2285
Thesaurus Paths	25.0%	.1977
Synonym	20.7%	.1890
Antonym	24.0%	.2142
Hypernym	22.7%	.1956
Hyponym	24.9%	.2030
Meronym:substance	20.0%	.2000
Meronym:part	20.8%	.2000
Meronym:member	20.0%	.2000
Holonym:substance	20.0%	.2000
Holonym:member	20.0%	.2000
Similarity:dict	18.0%	.2000
Similarity:wordsmyth	29.4%	.2058
all: mixture	42.0%	.2370
all: logarithmic	43.0%	.2354
all: product	45.0%	.2512
no PV: mixture	31.0%	.2135
no PV: logarithmic	30.0%	.2063
no PV: product	37.0%	.2207

Table 4: Comparison of results for merging rules on analogy problems.

We once again examined the result of deducting 1/2 point for each wrong answer. The full set of modules scored 31, 33, and 43 using the mixture, logarithmic, and product rules. As in the synonym problems, the logarithmic and product rules assigned probabilities more precisely. In this case, the product rule appears to have a major advantage, although this might be due to the particulars of the modules we used in this test.

The TOEFL synonym problems proved fruitful in spurring research into computational approaches to lexical semantics. We believe attacking analogy problems could serve the research community even better, and have created a set of 100 previously published SAT analogy problems (Claman 2000). Our best analogy solver from the previous experiment has an accuracy of 55.0% on this test set.[6] We hope to inspire others to use the same set of instances in future work.

[6]Although less accurate than our synonym solver, the analogy solver is similar in that it excludes 3 of the 5 choices for each instance, on average, while the synonym solver excludes roughly 3 of the 4 choices for each instance.

4 CONCLUSION

We applied three trained merging rules to a set of multiple-choice problems and found all were able to produce state-of-the-art performance on a standardized synonym task by combining four less accurate modules. Although all three rules produced comparable accuracy, the popular mixture rule was consistently weaker than the logarithmic and product rules at assigning high probabilities to correct answers. We provided first results on a challenging verbal analogy task with a set of novel modules that use both lexical databases and statistical information.

In nearly all the tests that we ran, the logarithmic rule and our novel product rule behaved similarly, with a hint of an advantage for the product rule. One point in favor of the logarithmic rule is that it has been better studied so its theoretical properties are better understood. It also is able to "sharpen" probability distributions, which the product rule cannot do without removing upper the bound on weights. On the other hand, the product rule is simpler, executes much more rapidly (8 times faster in our experiments), and is more robust in the face of modules returning zero probabilities. We feel the strong showing of the product rule on lexical multiple-choice problems proves it worthy of further study.

Acknowledgments

The research was supported in part by a grant from NASA. We'd like to thank the students and TAs of Princeton's COS302 in 2001 for their pioneering efforts in solving analogy problems; Douglas Corey Campbell, Brandon Braunstein, and Paul Simbi, who provided the first version of our Thesaurus module; Yann LeCun for scanning help, and Haym Hirsh, Rob Schapire, Matthew Stone, and Kagan Tumer who served as sounding boards on the topic of module merging.

References

Eric Brill and Jun Wu. Classifier combination for improved lexical disambiguation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 191–195, 1998.

David J. Chalmers, Robert M. French, and Douglas R. Hofstadter. High-level perception, representation, and analogy: a critique of artificial intelligence methodology. *Journal of Experimental and Theoretical Artificial Intelligence*, 4:185–211, 1992.

Cathy Claman. *10 Real SATs*. College Entrance Examination Board, 2000.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

Radu Florian and David Yarowsky. Modeling consensus: Classifier combination for word sense disambiguation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 25–32, 2002.

Robert M. French. The computational modeling of analogy-making. *Trends in Cognitive Sciences*, 6(5):200–205, 2002.

Tom Heskes. Selecting weighting factors in logarithmic opinion pools. In *Advances in Neural Information Processing Systems, 10*, pages 266–272, 1998.

Geoffrey E. Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN 99)*, volume 1, pages 1–6, 1999.

Robert A. Jacobs. Methods for combining experts' probability assessments. *Neural Computation*, 7(5):867–888, 1995.

Robert A. Jacobs, Michael I. Jordan, Steve J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of experts. *Neural Computation*, 3:79–87, 1991.

Mario Jarmasz and Stan Szpakowicz. *Roget's Thesaurus* and semantic similarity. Submitted for review, 2002.

George Lakoff and Mark Johnson. *Metaphors We Live By*. University of Chicago Press, 1980.

Thomas K. Landauer and Susan T. Dumais. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

Michael L. Littman, Greg A. Keim, and Noam Shazeer. A probabilistic approach to solving crossword puzzles. *Artificial Intelligence*, 134(1–2):23–55, 2002.

Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406, 1999.

Egidio Terra and C. L. A. Clarke. Frequency estimates for statistical word similarity measures. To be presented at the HLT/NAACL 2003, Edmonton, Alberta, May 2003, 2003.

Peter D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, pages 491–502, 2001.

Lei Xu, Adam Krzyzak, and Ching Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435, 1992.