ODNI-OUSD(I) Xtend Follow Up*

Machine Evaluation of Analytic Products: Detailed Description

Murray Miron

Contents

*The Office of the Director of National Intelligence, with the Office of the Under Secretary of Defense for Intelligence, sought ideas and descriptions of a viable technical approach for enabling the automated evaluation of finished intelligence products. This author's submission won that national Xtend contest, which prompted the ODNI-OUSD(I) to invite this follow up from me.

List of Algorithms

Part I. Front matter

1 References

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. `https://www.tensorflow.org/`.

[2] Ammar, W., Peters, M., Bhagavatula, C., and Power, R. The ai2 system at semeval-2017 task 10 (scienceie): semi-supervised end-to-end entity and relation extraction. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 592–596. `http://www.aclweb.org/anthology/S17-2097`.

[3] Augenstein, I., Das, M., Riedel, S., Vikraman, L., and McCallum, A. Semeval 2017 task 10: Scienceie - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 546–555. `http://www.aclweb.org/anthology/S17-2091`.

[4] Barik, B., and Marsi, E. Ntnu-2 at semeval-2017 task 10: Identifying synonym and hyponym relations among keyphrases in scientific documents. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 965–968. `http://www.aclweb.org/anthology/S17-2168`.

[5] Berend, G. Szte-nlp at semeval-2017 task 10: A high precision sequence model for keyphrase extraction utilizing sparse coding for feature generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 990–994. `http://www.aclweb.org/anthology/S17-2173`.

[6] Bethard, S., Savova, G., Palmer, M., and Pustejovsky, J. Semeval-2017 task 12: Clinical tempeval. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver,

Canada, August 2017), Association for Computational Linguistics, pp. 565–572. http://www.aclweb.org/anthology/S17-2093.

[7] Bies, A., Song, Z., Getman, J., Ellis, J., Mott, J., Strassel, S., Palmer, M., Mitamura, T., Freedman, M., Ji, H., et al. A comparison of event representations in deft. In *Proceedings of the Fourth Workshop on Events* (2016), pp. 27–36.

[8] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics 5* (2017), 135–146. https://arxiv.org/pdf/1607.04606.pdf.

[9] Buys, J., and Blunsom, P. Oxford at semeval-2017 task 9: Neural amr parsing with pointer-augmented attention. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 914–919. http://www.aclweb.org/anthology/S17-2157.

[10] Eger, S., Do Dinh, E.-L., Kuznetsov, I., Kiaeeha, M., and Gurevych, I. Eelection at semeval-2017 task 10: Ensemble of neural learners for keyphrase classification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 942–946. http://www.aclweb.org/anthology/S17-2163.

[11] Ghosal, D., Bhatnagar, S., Akhtar, M. S., Ekbal, A., and Bhattacharyya, P. Iitp at semeval-2017 task 5: An ensemble of deep learning and feature based models for financial sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 899–903. http://www.aclweb.org/anthology/S17-2154.

[12] Google. A comprehensive introduction to the word2vec algorithm, using tensorflow. https://www.tensorflow.org/tutorials/word2vec.

[13] Gruzitis, N., Gosko, D., and Barzdins, G. Rigotrio at semeval-2017 task 9: Combining machine learning and grammar engineering for amr parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 924–928. http://www.aclweb.org/anthology/S17-2159.

[14] Hernandez, S. D., Buscaldi, D., and Charnois, T. Lipn at semeval-2017 task 10: Filtering candidate keyphrases from scientific publications with part-of-speech tag sequences to train a sequence labeling model. In *Proceedings of the 11th International Workshop on Semantic Evaluation*

(SemEval-2017) (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 995–999. http://www.aclweb.org/anthology/S17-2174.

[15] Lee, J. Y., Dernoncourt, F., and Szolovits, P. Mit at semeval-2017 task 10: Relation extraction with convolutional neural networks. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 978–984. http://www.aclweb.org/anthology/S17-2171.

[16] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations* (2014), pp. 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

[17] Marsi, E., Sikdar, U. K., Marco, C., Barik, B., and Sætre, R. Ntnu-1@scienceie at semeval-2017 task 10: Identifying and labelling keyphrases with conditional random fields. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 938–941. http://www.aclweb.org/anthology/S17-2162.

[18] May, J., and Priyadarshi, J. Semeval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 536–545. http://www.aclweb.org/anthology/S17-2090.

[19] Mille, S., Carlini, R., Burga, A., and Wanner, L. Forge at semeval-2017 task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 920–923. http://www.aclweb.org/anthology/S17-2158.

[20] Mohamed, A. A., and Rajasekaran, S. Improving query-based summarization using document graphs. In *Signal Processing and Information Technology, 2006 IEEE International Symposium on* (2006), IEEE, pp. 408–410.

[21] Nguyen, K., and Nguyen, D. Uit-dangnt-clnlp at semeval-2017 task 9: Building scientific concept fixing patterns for improving camr. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 909–913. http://www.aclweb.org/anthology/S17-2156.

[22] P R, S., R, M., and Niwa, Y. Hitachi at semeval-2017 task 12: System for temporal information extraction from clinical notes. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 1005–1009. http://www.aclweb.org/anthology/S17-2176.

[23] Pivovarova, L., Escoter, L., Klami, A., and Yangarber, R. Hcs at semeval-2017 task 5: Polarity detection in business news using convolutional neural networks. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 842–846. http://www.aclweb.org/anthology/S17-2143.

[24] Saleiro, P., Mendes Rodrigues, E., Soares, C., and Oliveira, E. Feup at semeval-2017 task 5: Predicting sentiment polarity and intensity with financial word embeddings. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 904–908. http://www.aclweb.org/anthology/S17-2155.

[25] Schouten, K., Frasincar, F., and de Jong, F. Commit at semeval-2017 task 5: Ontology-based method for sentiment analysis of financial headlines. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 883–887. http://www.aclweb.org/anthology/S17-2151.

[26] Segura-Bedmar, I., Colón-Ruiz, C., and Martínez, P. Labda at semeval-2017 task 10: Extracting keyphrases from scientific publications by combining the banner tool and the umls semantic network. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 947–950. http://www.aclweb.org/anthology/S17-2164.

[27] Singh, V., Narayan, S., Akhtar, M. S., Ekbal, A., and Bhattacharyya, P. Iitp at semeval-2017 task 8 : A supervised approach for rumour evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 497–501. http://www.aclweb.org/anthology/S17-2087.

[28] Srivastava, A., Rehm, G., and Moreno Schneider, J. Dfki-dkt at semeval-2017 task 8: Rumour detection and classification using cascading heuristics. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 486–490. http://www.aclweb.org/anthology/S17-2085.

[29] Tourille, J., Ferret, O., Tannier, X., and Névéol, A. Limsi-cot at semeval-2017 task 12: Neural architecture for temporal information extraction from clinical narratives. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 597–602. http://www.aclweb.org/anthology/S17-2098.

[30] van Noord, R., and Bos, J. The meaning factory at semeval-2017 task 9: Producing amrs with neural semantic parsing. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 929–933. http://www.aclweb.org/anthology/S17-2160.

[31] Wang, C., Xue, N., and Pradhan, S. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Denver, Colorado, May–June 2015), Association for Computational Linguistics, pp. 366–375. http://www.aclweb.org/anthology/N15-1040.

[32] Wang, L., and Li, S. Pku_icl at semeval-2017 task 10: Keyphrase extraction with model ensemble and external knowledge. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (Vancouver, Canada, August 2017), Association for Computational Linguistics, pp. 934–937. http://www.aclweb.org/anthology/S17-2161.

[33] Yin, W., Yu, M., Xiang, B., Zhou, B., and Schütze, H. Simple question answering by attentive convolutional neural network. *arXiv preprint arXiv:1606.03391* (2016).

2 Personal note

I would like to thank the Office of the Director of National Intelligence, as well as the Office of the Under Secretary of Defense for Intelligence, for their sponsorship. Their investments in our country, our citizens, and our industries does not go unappreciated.

3 Foreword

I cannot overstate the complexity of the problem at hand. Sadly, no system in human history has managed to accomplish the goals of this challenge, at least not with the same level of proficiency as humans possess. A thorough treatment of all relevant algorithms would make this work thousands of pages; as such, links to implementations are often provided in place of original pseudocode.

I have endeavoured to produce something useful and informative, but certain assumptions had to be made regarding the form and content of analytic products. Small errors may be present.

4 Preface

What most people consider modern artificial intelligence can be more appropriately referred to as deep learning and/or machine learning. Examples of deep learning frameworks that are widely used in open source applications are TensorFlow[1], PyTorch, and Theano; any framework will bring with it a tremendous amount of implementation- and/or task-specific details that are beyond the scope of this work. Outside of provided references, no specific toolkits or languages are used. This has the benefit of being much more generic and widely applicable, but at the cost of leaving additional work for those who are concerned with one particular set of tools.

All deep learning techniques, including ensemble techniques[2], share a basic element, however: their foundation is an optimization problem. While it's common to think of artificial neural nets as digital brains, they are really – at their core – statistical models. What we refer to as "training" is really just the alteration of weight vectors in the model with respect to a given metric (often the logistical loss or cross-entropy). Because of this, all frameworks can be assumed to be differentiable throughout; this ensures that the error at any point in the model can be calculated.

The most popular method of representing language is, currently, to embed words within a multidimensional vectorspace.[3] They are generally a fundamental piece of any algorithm meant for computational linguistics, and are an area of research in themselves.

[1] TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc.

[2] Ensemble methods are those which use disparate machine learning methods together in a single implementation

[3] By the strict mathematical definition, they are not vectorspaces because they don't always conform to the constraints of a real vectorspace.

5 Primer

I would be remiss if I did not include a basic primer to aid the interested reader in exploring some of the less forgiving references herein. To that end, the most commonly used phrases in modern machine learning, as well as (what is hopefully) an intuitive explanation of their meaning, follows.

5.1 Definitions

- **Artificial neural network**: Abstractly, they are no different or more complicated than any standard graph, i.e. a set of nodes (usually depicted by filled-in circles) connected by a set of edges (the lines in the graph). The operations that are performed on a given piece of data as it passes along the edges, from node to node, are implementation specific and very quickly become quite complex: the effectiveness of modern neural networks lies primarily in the behavior of a set of relatively simple mathematical functions, e.g. sinusoidal functions (*sin*, *cos*, and so on.)

- **Activation function**: In almost all cases, at all nodes in a neural network (i.e. graph), an activation function (or gating function) is applied to determine whether or not input that has been given to a specific node (usually from prior layers of nodes) will be output to subsequent nodes in the graph. The activation function is applied to the values passed to a specific node. Conceptually, it is applied by the node in question itself. If the input is of a certain range of values, it will be passed on. If not, the input to the node is either not passed on at all, or a modified value is passed on instead. In the literature, it is extremely common to see the activation function being spoken of as though it were the node itself (frequently the activation function is the only property of the node we care about).

- **Rectified linear units (ReLU)**: a very intimidating name for a very simple thing. A ReLU is a node that employs a "rectified" linear activation function. It's nothing more than this: if the value of the input is less than 0, pass on 0 instead. If it's greater than 0, pass on the result of multiplying the input by whatever weight value is currently assigned to the edge in question. Conceptually, that is literally *all* the ubiquitous rectified linear unit means. ReLUs are used as the activation function for the vast majority of modern artificial neural nets, as their behavior under most circumstances is highly desirable (for reasons that are beyond the scope of this work).

- **Training**: the iterative process of modifying the weight values for all edges in the graph, such that for any input to the graph (neural network), the output received from the graph is as close to the desired values as possible. On a related note, this is why such immense processing power is necessary for modern machine learning techniques (we're talking about

trillions upon trillions of updates). Note that it is the *training* of the network that is so incredibly demanding. To actually *use* a neural network takes comparatively very little processing power.

- **Supervised training**: any method of training that employs hand-labeled examples. The output of a given neural network is compared to the labeled value, and the error between them is used to update the weights of every edge in the graph.
- **Unsupervised training**: any method of training which does not employ human labeled data; these methods are *usually*, but not always, much less effective than supervised training. They are also much more complex and beyond the scope of this work.
- **Optimizer**: a mathematical function or algorithmic process that is employed during training. It "optimizes" the network by specifying how the weights in a graph are updated at each step in the training process.
- **Stochastic gradient descent methods**: extremely common optimization methods that employ a technique involving partially random values when updating weights.

5.2 Crash course

In nearly all cases, input data is translated to a numeric representation, the result of which is given to a specific layer of nodes which are assigned the duty of being the input layer. Each node takes that value, and applies an activation function (usually ReLU) to determine what value it passes on to the nodes it is connected to. This process is repeated throughout the graph. Eventually, the output layer is reached, which is taken to be the output of the neural network.

Input layers, output layers, activation functions, optimization methods, the number of neurons (as well as the connections between them), and quite literally every other property of a model are chosen in whatever fashion will ensure the best performance for a given task.

There is no substitute for experience when it comes to designing models. Frequently, even the best practitioners are unable to predict what methods will yield the best performance. This is simply a fact of life when it comes to modern machine learning. To truly understand or apply modern artificial intelligence, all relevant information must be available, and a great deal of experience and education is required regarding the behavior of mathematical functions and implementation methods.

When it comes to deep learning, however, the basic conceptual information given above is enough to get at least something out of most papers. Advances in available hardware, as well as the lowered cost thereof, are among the reasons that people are so excited about deep learning in recent years (more and more possibilities are being opened up by cheaper hardware and improved designs).

Part II. Foundations of the method

6 Word2vec

Both the algorithm and project that are widely referred to as word2vec [1] are based on taking language and constructing vectors with which to work on, i.e. N-dimensional arrays to use as input to an artificial neural net. It was mentioned above, but is further illustrated here.

6.1 A simplified overview

Words that occur in similar locations and similar contexts within the sample end up with similar vector representations. A brief – and contrived – example of this can be found below. The vector representations are (one of) the objects that artificial intelligence models operate on.

The vast majority of computational linguistics systems are trained in a supervised fashion; id est, they are trained with labeled datasets that are compiled manually by humans for that purpose[4]. A given model is then optimized (trained) so that the output shows two properties: one, it assigns labels that are substantially similar in nature to the human assigned labels; and two, the similarities are not so extreme that it fails to be useful when given input it has never seen before.

In other words, if we think of the trained model as a mathematical function

$$f(x) \rightarrow y$$

which takes input x and yields output y, then it must generalize to values of x that it hasn't seen before. It must also produce output values of y, for a given set of inputs x, that are close enough to the "true" function to be useful.[5]

Training a model beyond this useful point is called "overfitting" (when using the term overfitting colloquially, at any rate.) It comes from the model too closely "fitting" the curve of the training data, thereby losing generality and thus losing usefulness.

For a more rigorous and comprehensive overview, see:

`https://www.tensorflow.org/tutorials/word2vec`

Note the Creative Commons Attribution 3.0 license for the tutorial, and the Apache 2.0 license for the code.

6.2 Some simplified examples

Let us consider how processing a truly arbitrary sentence might influence the resulting vectors.

[4] Captchas are an example of crowdsourcing this information, and it is why the large players in the industry have the upper hand (Google does more than just sell user data).

[5] The "true" function is the unknown one that we seek to "train" our model to closely approximate.

```
The cat slept in the hat.
```

Will be translated to vector representation, where each word will end up being stored as something similar to:

```
[0.9 0.8 1.7 2.3]
```

Each word (or token) will be a unique vector, i.e. a N-dimensional array (matrix). The values of the elements in the vector are constructed based on the words which surround the token being processed. This ensures that similar words have similar vector values. For example, the sentences:

```
I went to Spain.
```

And

```
I went to Greece.
```

Would cause *Greece* and *Spain* to have relatively similar values in their vector representations; this allows a great deal of the implicit information that's contained within language to be stored.

Let us add another sentence to our contrived example. Consider the independent and unrelated sentence which follows:

```
Highschools have unusually long summer vacations in America.
```

After processing all three example sentences, the vector representations for *America* and *Spain* would show no significant similarities. That might change, however, if additional sentences used the words in similar contexts later on.

7 FastText

Facebook, inc. recently open sourced one of their in-house tools. FastText [8] is a small C++ program designed primarily for text classification. Unlike entire deep learning frameworks that serve only the developer community (e.g. TensorFlow), FastText [8] aims to be a tool for advanced users: it implements several of the more recently popular techniques.

8 CoreNLP

Stanford's CoreNLP [16] is an excellent implementation of some of the more mundane functions that are assumed to be available in the pseudocode that follows; see https://stanfordnlp.github.io/CoreNLP/ [16] for details and working code.

Part III. Solution

9 Evaluation criteria

9.1 "Properly describes quality and credibility of underlying sources, data, and methodologies"

A complex but effective method for abstract meaning representation (AMR) that was proposed by Wang, Xue, and Pradhan [31] is assumed; an implementation of this algorithm can be found at https://github.com/c-amr/camr. Note that the project is licensed under the GNU General Public License v2.0 terms; see the URL provided for further details.

The basic idea behind algorithm 1 is to step through the AMR graph of an analytic product looking for singular references. To that end, an adjacency matrix for the transition-based construct described in [31] is used. Note that to this author's knowledge, that team's last research report states an F-Score of 0.61 on a blind test dataset; the maximum F-Score is 1, and the minimum is 0. One could also use the terminal states generated by the transition-based dependency parser while constructing the AMR for a given analytic product, or even the dependency tree as opposed to the graph.

The transition-based method employed is the key to this actually working, since singular references to an abstract meaning representation – i.e. an orphaned node in the dependency tree, AMR graph, or transition rules – would not be meaningful without it.

Scoring algorithm 1 Properly describes quality and credibility of underlying sources, data, and methodologies

```
1:  procedure SCORE
2:      product ← the analytic product in question
3:      amr ← output of Wang, et al [31] when given product
4:      score ← maximum score
5:      x ← adjacency matrix for amr                          ▷ 5
6:      for all {x_{i,j} | x_{i,j} > 0} do
7:          if (x^2)_{i,j} < 1 then                           ▷ 7
8:              score ← score − 1
9:      if score < 0 then score ← 0
10:     return score
```

5. The terminal states S_t of the dependency parser S can also be used with minimal differences in the iterative loop.

7. Note that multiplying an adjacency matrix by itself yields a new matrix where the (i,j) elements are the existence of paths, of length 2, from i to j.

9.2 "Properly expresses and explains uncertainties associated with major analytic judgments"

This is one of the trickier requirements to satisfy; the algorithm 1 is modified and extended. I would like to reiterate Wang, et al's [31] F-score: this algorithm will work, but there will be frequent mistakes.

The call to *sentiment_analysis()* is assumed to return a negative value for undesirable sentiment, a positive value for desirable sentiment, and a 0 otherwise.

Scoring algorithm 2 Properly expresses and explains uncertainties associated with major analytic judgments

```
1: score ← maximum score
2: product ← the analytic product in question
3: amr ← output of Wang, et al [31] when given product
4: ambiguous ← 0
5: procedure SCAN(amr)
6:     x ← adjacency matrix for amr
7:     for all {x_{i,j} | x_{i,j} ≥ 1} do
8:         if (x^2)_{i,j} < 1 then
9:             ambiguous ← (x^2)_{i,j}
10: procedure SCORE(ambiguous)
11:    for all {y | y ∈ ambiguous} do
12:        if CALL(sentiment_analysis)(y) = 0 then
13:            score ← score − 1
14: if score < 0 then score ← 0
15: return score
```

9.3 "Properly distinguishes between underlying intelligence information and analysts' assumptions and judgments"

Upon further investigation, the method that was originally suggested to fulfill this requirement was found to be no better than tossing a fair coin. No other means of accomplishing this evaluation criteria was found within the required timeframe.

Additionally, I would like to point out that this requirement may be literally impossible to satisfy; in a very similar manner to Einstein's special relativity and absolute frames of reference, absolute facts simply do not exist. Even deciding whether one event takes place before, or after, another event is no simple matter in the grand scheme of things: among other less practical complications, there's the issue of relativity and time/length contraction.

In short, I'm not familiar enough with intelligence products to properly contrast underlying intelligence information with an analysts' assumptions or judgments; still, I feel that emulating typical human intuitions regarding this evaluation criteria should be the focus here.

9.4 "Demonstrates relevance to customers and addresses implications and opportunities"

I can think of no way to address this solution requirement without further, and always up-to-date, information; relevance does change as time passes, after all.

9.5 "Uses clear and logical argumentation"

This is a very easy evaluation criteria to satisfy: the transition rules crafted as part of algorithms 1 and 2 inherently offer an evaluation of this.

Scoring algorithm 3 Uses clear and logical argumentation

```
1: score ← maximum score
2: product ← the analytic product in question
3: amr ← output of Wang, et al [31] when given product
4: procedure SCORE
5:     for all errors ∈ AMR buffer do
6:         score ← score − 1
7:     if score < 0 then
8:         score ← 0
9:     return score
```

9.6 "Explains change to or consistency of analytic judgments"

Here we have the same problem that was specified in evaluation criteria 9.3. What constitutes an analytic judgment, as contrasted with underlying intelligence information, and what is the domain of concern with respect to said judgments (that analyst, the source country, the world, or something else)?

9.7 "Makes sound judgments and assessments"

Even IBM's Watson cannot hope to satisfactorily evaluate the range of judgments that are likely to be found in intelligence products. I cannot claim to have beaten IBM, regrettably; however, if the Xpress challenge is fruitful and a predictable structure in intelligence products could be assumed, this would become trivial.

9.8 "Incorporates effective visual information where appropriate"

A method which will work in many cases is a fully convolutional network with attention mechanisms, trained to allow for posing short questions. [33] The short questions themselves come from the AMR graph.

This method will be very hit-or-miss, and to compensate for that, only a binary score will be possible (i.e. "is the image relevant, or is it just not useful at all," basically). Distinguishing between cases of poor information, and cases where visual aids are necessary to convey information, is something even we humans do very poorly.

Scoring algorithm 4 Incorporates effective visual information where appropriate

```
1: product ← the analytic product in question
2: amr ← output of Wang, et al [31] when given product
3: network ← pretrained network, as specified
4: procedure SCORE(product)
5:     for all questions ∈ AMR buffer do
6:         if posing question to network yields an answer then
7:             return GOOD
8:     return BAD
```

10 Prototype development

10.1 Resources

There is absolutely no way around the need for a very large number of analytic products, as well as the evaluations thereof. They need not be sensitive or classified, and they need not be current; they do need to be readily available, however. They should also be a representative sample of the types of intelligence products the system will be asked to evaluate. A corpus of thousands of examples would be required; more is always better. The thing to remember here is that the system will be trained to evaluate products which bear a resemblance to the examples it has seen, in the way that the evaluations it has seen scored said products. Without a corpus that covers the possible permutations, the system would not work well at all.

There are a great many people who could provide a poorly working system to perform the task at hand. One that works well enough to be worth using, however, always requires domain specific knowledge – in this case, knowledge of intelligence products and the lifecycle thereof – as well as the sort of expertise that is uncommon. The best systems always require technical expertise and personal knowledge of the domain.

All resources and requirements except the large set of examples are flexible or negotiable, but to actually install and/or maintain the system, a very trust worthy entity seems necessary: if the developers don't have first hand knowledge of what the system will be dealing with, the system cannot be designed to deal with it. It's just that simple.

10.1.1 A word to the wise

One of the areas that IBM has marketed their Watson system to is healthcare administration. A certain hospital (which shall remain nameless) contracted with IBM. After approximately a year or more, and through no fault of IBM, their system was in a state of utter uselessness.

The users of it refused (or simply did not care) to follow necessary protocols, and IBM engineers were not given the level of access, or did not have enough hours in their days, to maintain the system. What resulted was the loss of several management paychecks and at least one administrator. I do not know the details of what came after that.

The more complex the tool, the more protocols for use and upkeep that are required; we are not talking about pocket calculators.

10.2 Timeframe

To develop the system, it would take much longer than 60 days. At a bare minimum, six months; considering real world concerns, two years seems likely.

10.3 Costs

It's not the sort of thing you want to go with the lowest bidder on, I'll put it that way. To be frank, given the cutting edge nature of the task at hand compounded with the supercomputer level processing power necessary for training (i.e. very expensive distributed computing systems), going with one of the big boys (e.g. IBM or Google), if they were amenable, would probably be the best bet.

IBM or Google would likely make costs exorbitant. By way of example, IBM is estimated as having spent $900 million dollars during Watson's initial three year development period (the kind of talent required to make this work well does not come cheap).