Deep Learning for Automatic Summary Scoring

Pierre-Etienne Genest[1], Fabrizio Gotti[1], and Yoshua Bengio[2]

Université de Montréal
Département d'informatique et de recherche opérationnelle
Laboratories RALI[1] and LISA[2]
{genestpe,gottif,bengioy}@iro.umontreal.ca
rali.iro.umontreal.ca, www.iro.umontreal.ca/~lisa

Abstract. Automatic summary scoring is used very often by summarization system developers to test different algorithms and to tune their system. We have developed a new approach based on representation learning, using both unsupervised and supervised learning components, to score a summary based on examples of manually evaluated summaries. Our deep learning approach greatly surpassed ROUGE in terms of correlation with Pyramid (content) scores for individual summaries. However, ROUGE performed slightly better when comparing summarization systems based on their average score.

1 Introduction

Progress in the field of Text Summarization requires ways of assessing the quality of summaries. Comparisons and ranking of summarization systems in international evaluations like the Text Analysis Conference (TAC) [7] rely first on manual summary scoring. On the other hand, automatic summary scoring is used very often by summarization system developers to test different algorithms and to tune their system. Although automatic summary scoring is not as reliable, it does not require human resources nor a long time to complete, so it can be repeated as often as necessary.

The best known and most trusted automatic metric for summary evaluation is the so-called Recall-Oriented Understudy for Gisting Evaluation, or ROUGE [16]. This metric is strictly based on n-gram similarity scores between a model summary and the summary to be evaluated.

In this paper, we describe a new approach to automatic summary scoring based on representation learning, using both unsupervised and supervised learning components, to score a summary based on examples of manually evaluated summaries. This is based on recently introduced algorithms for deep learning of representations [12, 11, 1], and is based on a novel architecture for comparing the learned representations associated with two preprocessed summaries, in the spirit of so-called Siamese Networks [5]. Like ROUGE, it also relies on a comparison between a model summary and the summary to be evaluated. This is accomplished in three steps. First, we preprocess the summaries so that they can be expressed as a vector in term-space. The second step attempts to learn

a mapping from the term-space into a concept-space representation of much smaller dimensionality, using an unsupervised auto-encoder [19, 4, 11] trained on a large corpus. That learned intermediate lower-dimensional representation is more abstract and more oriented towards semantics than the raw term-space representation, because combinations of words that have a similar meaning tend to be represented by nearby vectors in that space, in a way similar to Latent Semantic Analysis [8]. Finally, the third step stacks a regression neural network on top of the attribute-wise comparison obtained from the concept-space representations of the summary to be evaluated and of the model summary.

An important advantage of many Deep Learning algorithms is that they can exploit large quantities of unlabeled data to learn better representations [1], that can generally be more easily transferred across different domains [2] or combined with labeled data for semi-supervised learning [24]. The basic hypothesis explaining these earlier successes [9] is that for the type of tasks at hand (and presumably most tasks considered for AI), representations $h(x)$ of inputs $X = x$ that are useful at characterizing $P(X)$ are useful at characterizing $P(Y|x)$ (where Y's are target labels to predict).

Section 2 will describe existing summary evaluation metrics. We give the details of our approach, including how we implemented it, in section 3. Section 4 discusses our results, and we conclude in section 5.

2 Summary Evaluation Metrics

2.1 Direct Manual Metrics

Direct manual metrics produce human-made scores given by subjective criteria. They are generally scaled by integers between 1 and 5 or 1 and 10. The most common, called Overall Responsiveness in the TAC conferences, answers a question such as "Is this a good summary of the document(s)?" and "How much would you pay for this summary?". The other common measure is a linguistic quality score, which assesses a summary's grammaticality, as well as its focus, coherence, etc.

2.2 Pyramid

The Pyramid metric [17] is an indirect manual evaluation metric of a summary's content. Human assessors read each model summary and determine each one's Semantic Content Units (SCUs) – the ideas or statements of a text. The Pyramid content score of a summary to be evaluated is given by the recall of model SCUs present, weighed by the number of model summaries that contained each SCU, if more than one model was available. The Pyramid score is the one we attempt to predict with our approach.

2.3 ROUGE

ROUGE [16] is an automatic evaluation metric that computes an n-gram similarity score between the model summary and the summary to be evaluated. Several

types of ROUGE measures exist, and the one with the highest correlation with manual scores is ROUGE-2 recall – the recall of model summary bigrams. Very high correlations between manual metrics and ROUGE have been observed [6].

3 Our Approach

This section has five subsections as follows. First, we describe the data sets used for training and testing. Then, we describe the three steps of our approach, namely preprocessing the documents into a binary vector in term-space, learning a representation of this term-space into a concept-space, producing a comparative vector (comparing the attributes of the two summaries), and training a neural network to predict the Pyramid score. These are described at a high level, with the last subsection giving technical details of the implementation.

3.1 Data Sets

Most machine learning approaches require large data sets to perform well, and examples of manually scored summaries are relatively rare. The TAC conferences offer a good source of such data and we used the 2008 and 2009 sets because they were recent, had short (100-word) summaries, and were both done using the same task description, namely query-focused, multi-document, 100-word summaries. Together, they contained roughly 4,000 manually evaluated summaries (excluding update summaries which respond to a different task), completed on 92 document sets. The Pyramid scores of these summaries were normalized to be in the range [0.001, 0.999] for easier use by our algorithm.

The documents for which those summaries were written are NewsWire articles found in the Linguistic Data Consortium's AQUAINT-2 corpus [23], which contains more than 900,000 articles from six different news agencies. The autoencoder is trained using this corpus.

3.2 Preprocessing

It is desirable to represent data by its meaningful features for input to a machine learning algorithm. An easy way to do this for text is to use the so-called bag-of-words approach of representing a text by the terms it contains, regardless of the number of times each term occurs.

In this formalism, each document is represented by a binary vector of size equal to the number of terms in a given vocabulary. Each term in the vocabulary has an index in the vector, which corresponds to a dimension in this vector-space, or term-space. A document is thus a point in term-space, defined as a vector with 1's in dimensions corresponding to terms it contains, and 0's in all other dimensions.

Not all terms are important, however, and it is computationally impracticable to deal with a very large vocabulary, so we had to significantly reduce its size. First, the Porter Stemmer [18] is used on the terms, to represent identically

all terms of the same family. A vocabulary of 850,000 unique stems was initially found in the AQUAINT-2 corpus. All numbers and amounts were projected onto a single tag, and all terms that contain special characters were removed, taking care of a large portion of this number. Stop-words were also taken out, in order to prevent the noise the most common English words would likely create. Finally, we kept only the 10,000 most frequently occurring stems in the AQUAINT-2 corpus, a number high enough to cover terms from a wide range of subjects. This corresponds to terms with a minimum frequency of at least 2,500 within the 900,000 articles of the corpus.

3.3 Deep Learning and Auto-Encoder for Dimensionality Reduction

One of the basic ideas behind Deep Learning algorithms [1] is to exploit unsupervised learning to learn intermediate representations that can then be used in a supervised learning framework. In our work, an auto-encoder [11] is used to reduce the dimensionality of the term-space vectors, so that they can be represented in a much smaller concept-space. Auto-Encoders are a special type of multi-layer neural network with a hidden layer of small dimensionality, which represents an encoding that we are trying to learn. In our case, the encoding can be interpreted as a concept-space, made of learned non-linear transformations of the term-space, which can express the most essential features of a document. The encoding is learned by using the large AQUAINT-2 corpus for input. An important characteristic of such representation-learning algorithms is that they can exploit large quantities of unlabeled data. When they are trained on a bag-of-word, they learn a distributed semantic representation for each word [20], in which semantically similar words (or bags-of-words) are associated to nearby vectors.

The process is illustrated in Figure 1. The encoding function $\mathbf{f}$ is applied to the term-space vector $\mathbf{x}$ representing a corpus document in order to produce the concept-space vector $\mathbf{y}$. $\mathbf{f}$ consists of a linear transformation followed by a non-linear function. Next, function $\mathbf{g}$ decodes $\mathbf{y}$ back to a vector in term-space by applying again a linear transformation followed by a non-linear function. The auto-encoder is trained with the goal of learning the parameters of functions $\mathbf{f}$ and $\mathbf{g}$, so that the reconstructed vector $\tilde{\mathbf{x}}$ is similar to the original vector $\mathbf{x}$.

To assess the quality of the encoding and decoding, a loss function $\mathbf{L}$ matched to the non-linearity compares $\mathbf{x}$ and $\tilde{\mathbf{x}}$. The parameters of the auto-encoder are set to minimize the reconstruction error on the training data.

Because the input vectors are binary and sparse (95% of the dimensions contain zeros), we apply a sampling algorithm to speed up the processing. All dimensions that contain ones are sampled, as well as the same number of dimensions containing zeros, selected randomly. Reconstruction and error gradients are only computed for the sampled dimensions, i.e., no gradient is back-propagated for the non-sampled dimensions. To our knowledge, this sampling mechanism is novel in the context of auto-encoders, and it allowed us considerable speed-up of training time, around 8-fold. This is a major advantage to explore such un-

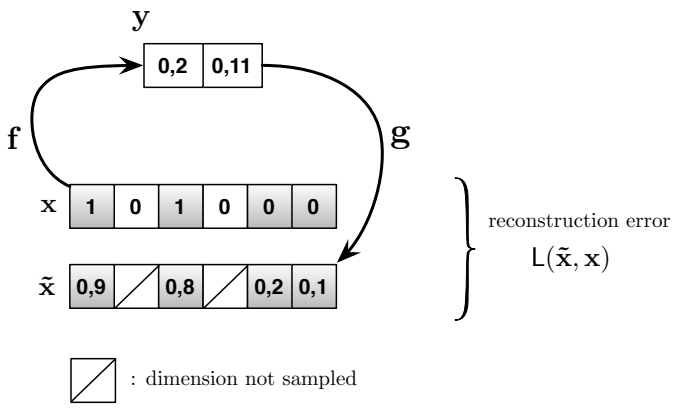

Fig. 1. Architecture of the auto-encoder to learn an encoding **y** in concept-space of an input **x** in term-space.

supervised learning algorithms in the context of the large number of training examples used (around a million).

Four values for the number of dimensions of the concept-space were tested: 200, 400, 600 and 1,000. A lower average reconstruction error for the auto-encoder, as well as a better performance when this encoding is used as part of the regression process were observed when the input is encoded in a concept-space of 600 dimensions.

Although the reconstruction error criterion of an auto-encoder does not correspond to training a probabilistic model of the input vectors, a slight modification of it, called the denoising auto-encoder [22], and in which the auto-encoder takes a corrupted input and tries to reconstruct the original clean input, does. A variant of the denoising auto-encoder corresponds [21] to applying a regularized Score Matching criterion [13] to a particular Restricted Boltzmann Machine [11, 12]. In our experiments we found that the addition of this corruption process helped, but only marginally, so results with the simpler ordinary auto-encoder are reported here.

3.4 Supervised Regression on Top of Learned Representation

The objective is to compare a summary **s** with a model summary **m**. On top of the representation learned by the auto-encoder for the summary **s** and for the model summary **m**, we first compute a "comparison" layer that performs an element-wise comparison between the two summaries' concept-space attributes, on top of which we then stack a supervised multi-layer regression neural network, to predict the summary Pyramid score p. The layout of the network is illustrated in Figure 2.

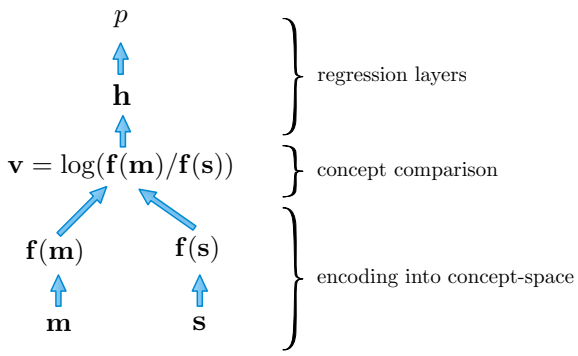

Fig. 2. Deep Learning architecture for regression, with three steps: encoding into concept-space, concept comparison and regression layers.

The input layer contains the two preprocessed summaries to be compared, represented by term-space vectors. The encoding function $\mathbf{f}$ learned by the auto-encoder is applied on both term-space vectors to reduce them to their concept-space representation.

The layer $\mathbf{v}$ compares the values of $\mathbf{f}(\mathbf{s})$ and $\mathbf{f}(\mathbf{m})$ for each concept (how strongly each concept is present in the summary and the model). For each dimension in concept-space, $\mathbf{v}$ is computed with the element-wise logarithm of the ratio of $\mathbf{f}(\mathbf{s})$ and $\mathbf{f}(\mathbf{m})$.

The remainder of the architecture is a standard one-hidden layer neural network for regression. The size of the hidden layer $\mathbf{h}$ was set to 1,000, because a much higher number of hidden units would have entailed a large increase in computing time yielding meager gains in performance.

Architecture variants. Four variants of the architecture have been considered, in order to test two hypotheses. Firstly, the impact of the auto-encoder unsupervised pre-training was tested. The function $\mathbf{f}$ of the architecture would either be initialized randomly, or by the learned function of the auto-encoder in order to test this hypothesis. Secondly, we tested whether or not the parameters of the concept-space encoding function $\mathbf{f}$ should be adapted during training or left untouched. Adjusting the parameters of $\mathbf{f}$ during training severely slows down the execution. Our experiments showed that there is always a very significant gain to use the auto-encoder-learned concepts rather than random ones, even when those can be adjusted during training. Also, statistical tests failed to observe a significant difference between keeping the learned concepts fixed and adjusting them to new data during training. These findings are different from those of previous work using auto-encoders in deep architectures [3, 14], which observed better results when all the parameters could be adjusted to the data.

3.5 Technical Details about the Implementation

Hyper-parameters For both the auto-encoder and the regression component, we tested several values for each of the hyperparameters of the algorithms, using a grid search. These include the size of the hidden layers and variations of the algorithms themselves, as already mentioned, but also the learning rate for gradient descent, its rate of decay and the number of iterations for early stopping. The gradient descent learning rate was taken as $\epsilon_t = \frac{\epsilon_0 \tau}{t+\tau}$, where ϵ_0 is the initial learning rate and τ controls the rate of decay (asymptotically in $1/t$, to guarantee convergence).

Auto-Encoder The two layers of the network are computed using $\mathbf{y} = \mathbf{f}(\mathbf{x}) = \tanh(\mathbf{W}\mathbf{x} + \mathbf{b})$ and $\tilde{\mathbf{x}} = \mathbf{g}(\mathbf{y}) = \sigma(\mathbf{W}'\mathbf{y} + \mathbf{b}')$, where σ is the logistic sigmoid. Therefore, $\mathbf{f}$ is defined by a linear transformation matrix $\mathbf{W}$ and two bias vectors $\mathbf{b}$ and $\mathbf{b}'$. We used tied weight matrices, so that $\mathbf{W}' = \mathbf{W}^T$. The loss function used is a cross-entropy between the reconstruction and the original vector, given by $\mathsf{L}(\tilde{\mathbf{x}}, \mathbf{x}) = -mean(\mathbf{x}\log(\tilde{\mathbf{x}}) + (1-\mathbf{x})\log(1-\tilde{\mathbf{x}}))$ where the mean is taken over the elements of $\mathbf{x}$. The matrix $\mathbf{W}$ is initialized randomly and uniformly, such that $W_{ij} \in [-\frac{1}{\sqrt{\bar{u}}}, \frac{1}{\sqrt{\bar{u}}}]$, with $\bar{u}$ the average number of ones in $\mathbf{x}$. The bias vectors were initialized with $\mathbf{b} \in [0, 2]$ and $\mathbf{b}' \in [-0.5, 0.5]$. These values were chosen to stay close to inflection points of the non-linear activation functions of each layer, as suggested in LeCun [15].

Comparison Layer and Regression Component The equation used for the comparison layer is slightly different from that in Figure 2, to avoid taking the logarithm of zero:

$$\mathbf{v} = \log\left(\frac{\mathbf{f}(\mathbf{m}) + 1 + \epsilon}{\mathbf{f}(\mathbf{s}) + 1 + \epsilon}\right), \tag{1}$$

where ϵ is 10^{-3}. The regression layers are computed similarly to $\mathbf{f}$ and $\mathbf{g}$. The hidden units $\mathbf{h}$ are computed by taking a linear transformation of $\mathbf{v}$ and applying a hyperbolic tangent. The score p was computed by a linear transformation of $\mathbf{h}$ and applying a logistic sigmoid activation function. The loss function is the Kullback-Liebler divergence between the predicted score and the target Pyramid score. The initialization of each matrix element for all layers was in the uniform interval $\left[-\sqrt{\frac{6}{n_i+n_j}}, \sqrt{\frac{6}{n_i+n_j}}\right]$, with n_i and n_j the number of units of the layer below and the current layer. All biases were initialized to 0.

4 Results and Analysis

4.1 Algorithmic Performance

Auto-Encoder The encoder was trained on 800,000 documents randomly selected from the 900,000 articles contained in the ACQUAINT-2 corpus. The

Correlation with Pyramid scores			
	Pearson	Spearman	Kendall
Deep Learner	0.786	0.782	0.591
ROUGE-2	0.617	0.591	0.427

Table 1. Correlation coefficients between our Deep Learning approach and Pyramid, and between ROUGE-2 and Pyramid ($n = 2288$), for individual summary predictions (comparing a single pair of summaries).

remaining 100,000 documents were set aside for testing purposes. The reconstruction error achieved by the auto-encoder on this test set is a cross-entropy of 0.134. This could for example be achieved by predicting (on average) a value of 0.13 for the dimensions of terms that did not appear in the input document, and predicting 0.87 for the dimensions of terms that did appear in it.

Regressor An 11-fold cross-validation was performed to evaluate the quality of the regression. A single fold consists of a training set including all of the 2008 TAC data and $10/11^{\text{th}}$ of the 2009 data. The remaining $1/11^{\text{th}}$ of the 2009 data constitutes the test set.

The mean absolute error between the predicted score for a summary and the actual Pyramid score in the test set is 0.085. Note that, although the scores are between 0 and 1, most of them are fairly low, with a mean of 0.25 and a standard deviation of 0.17, as can be observed in Figure 3 below.

4.2 Intrinsic Evaluation

In the language of Galliers and Spärck Jones [10], an intrinsic evaluation is related to a system's objective, whereas extrinsic evaluation is interested with a system's actual function. Because our objective is to simulate the Pyramid scores for evaluating summaries, it is interesting to compute correlation coefficients between our predictions and Pyramid scores of the test set. These correlation coefficients serve here as an intrinsic evaluation method. Table 1 shows our correlation with Pyramid and compares it with ROUGE's. Figure 3 plots all the data points for our predictions and ROUGE's.

These results are very satisfactory, as they show that our approach tends to rank summaries according to their content much better than ROUGE does. Although the correlation coefficients for individual comparisons are not very high compared to the correlation coefficients for a whole system (Table 2), they show a decent level of performance for an automatic evaluation metric. More importantly, there is a surprisingly large improvement in individual summary assessment when compared to ROUGE, a 32% relative improvement in Spearman correlation, going from .591 to .782 (see Table 1).

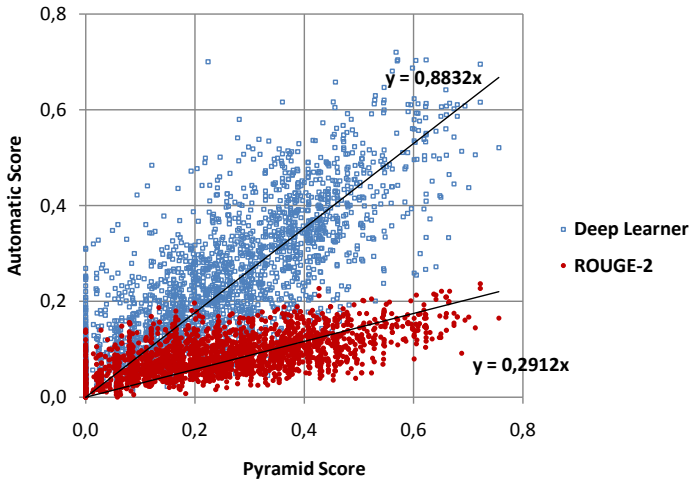

Fig. 3. Plot of the predictions from our Deep Learning approach and ROUGE-2 against Pyramid scores, for each of the 2288 summaries evaluated for testing. A large proportion of points have a Pyramid score of 0.

Correlation coefficients between system averages			
	Pearson	Spearman	Kendall
Deep Learner	0.946	0.898	0.751
ROUGE-2	0.972	0.942	0.803

Table 2. Correlation coefficients between per-system averages ($n = 52$).

4.3 Extrinsic Evaluation

In practice, evaluation metrics are not used to rank single summaries, but systems, or different configurations of a system. This is where automatic metrics like ROUGE are most useful: they allow the fine-tuning of a system, possibly iteratively, without having to manually evaluate the many summaries produced, or to even read any of them. The 2009 TAC data comprised summaries written by 52 automatic systems, so we averaged the scores for each system, over the 44 document sets. This was done for both our approach, Pyramid and ROUGE.

As table 2 shows, ROUGE averages are more correlated to Pyramid averages than our approach's averages are. That is, given many examples of two summarization systems's output, ROUGE-2 predicts slightly more accurately which one is better on average than our approach does. Figure 4 shows the actual data points for each system.

Moreover, this extrinsic discriminative power can be evaluated directly using Welch t-tests between pairs of systems. This is a way to verify how our approach and ROUGE rank any pair of systems, as compared to how Pyramid ranks them,

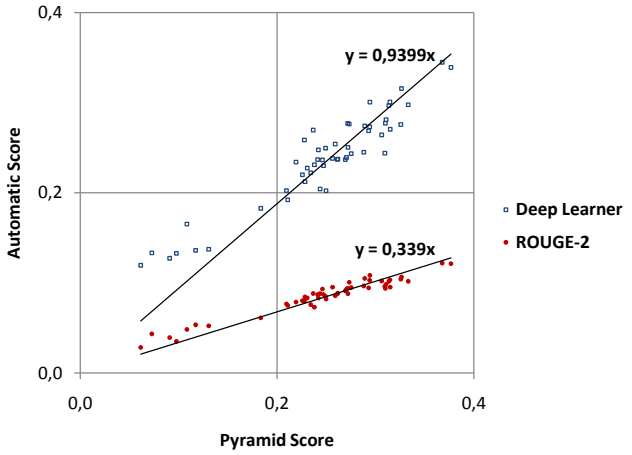

Fig. 4. Plot of averages of the predictions from our Deep Learning approach and ROUGE-2 against Pyramid scores, for each of the 52 evaluated systems.

Discriminative power between systems			
	Agreements	Disagreements	Contradictions
Deep Learner	1107	219	0
ROUGE-2	1124	202	0

Table 3. Number of agreements, disagreements and contradictions between an approach and Pyramid, from Welch t-tests conducted over all possible pairs of summarization systems.

using appropriate statistical tests. From two distributions A and B of predicted scores on two summarization systems, we verify if we observe a statistically significant difference between them using a Welch t-test. The same is done with the distribution of Pyramid scores. An agreement is when both tests produce the same result, namely that A and B are indistinguishable, that A is greater than B or the opposite. A disagreement is when one test shows no significant distinction while the other believes there is one. Finally, a contradiction occurs when one test shows that A is better than B and the other shows the opposite.

Ultimately, it is difficult to understand why our system has better intrinsic performance than ROUGE but slightly inferior extrinsic performance. There could have been some sort of noise-canceling effect for ROUGE that is less present with our approach.

5 Conclusion

We have introduced a way to speed-up training of unsupervised auto-encoders that learn a semantically beneficial representation of concepts from the given

term-space input, based on a sampling approximation of the training criterion. We have shown that an implementation of Deep Learning regression for summary evaluation scores can substantially improve on ROUGE at determining which of two summaries is better, yielding a 32% relative improvement in Spearman correlation, going from .591 to .782. On the task of determining which of two systems are better, and using only a simple averaging aggregation, performance was slightly worse than ROUGE-2 recall. This slightly inferior performance on per-system averages might come from the overspecialization of using a machine learning approach. The (intrinsic) goal that we set out for our approach was to predict Pyramid scores as best as possible for single observations. No specific learning of how to score systems given many example outputs was made, even though it is what most summarization developers actually need. With better assessment at the individual summary level, we believe that there is a great potential to extend these results to the system-level by better aggregation and training methods focused on improving the system-level predicted performance. For this purpose, it would be interesting to explore ways to design a machine learning algorithm which takes a set of input summaries and outputs something better than averaging the score predictions. This could be done by adding an extra layer to the architecture, or by running another algorithm to learn to combine scores efficiently for ranking purposes.

Another variant which should be explored is the insertion of an absolute value computation after the logarithm in equation 1. This would guarantee that the regression is invariant to switching the order of its two input summaries m and s, and $\mathbf{v}$ would measure the absolute discrepancy between various aspects of the summaries captured by each of the dimensions of $f(m)$ and $f(s)$.

References

1. Y. Bengio. Learning deep architectures for AI. *Foundations & Trends in Mach. Learn.*, 2(1):1–127, 2009.
2. Yoshua Bengio, Frederic Bastien, Arnaud Bergeron, Nicolas Boulanger-Lewandowski, Youssouf Chherawala, Moustapha Cisse, Myriam Côté, Dumitru Erhan, Jeremy Eustache, Xavier Glorot, Xavier Muller, Sylvain Pannetier-Lebeuf, Razvan Pascanu, François Savard, and Guillaume Sicard. Deep self-taught learning for handwritten character recognition. NIPS*2010 Deep Learning and Unsupervised Feature Learning Workshop, 2010.
3. Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In Bernhard Schölkopf, John Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 153–160. MIT Press, 2007.
4. Herv Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.
5. J. Bromley, J. Benz, L. Bottou, I. Guyon, L. Jackel, Y. LeCun, C. Moore, E. Sackinger, and R. Shah. Signature verification using a siamese time delay neural network. In *Advances in Pattern Recognition Systems using Neural Network Technologies*, pages 669–687. World Scientific, Singapore, 1993.

6. Hoa Trang Dang and Karolina Owczarzak. Overview of the TAC 2009 Summarization Track. In *Proceedings of the Second Text Analysis Conference*, Gaithersburg, Maryland, USA, 2009. National Institute of Standards and Technology.
7. Hoa Trang Dang and Karolina Owczarzak. Overview of the TAC 2010 Summarization Track. In *Proceedings of the Third Text Analysis Conference*, Gaithersburg, Maryland, USA, 2010. National Institute of Standards and Technology.
8. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
9. Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, February 2010.
10. J.R. Galliers and K. Spärck Jones. Evaluating Natural Language Processing Systems. Technical Report UCAM-CL-TR-291, U. of Cambridge, Computer Laboratory, 1993.
11. G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313:504–507, July 2006.
12. Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.
13. Aapo Hyvärinen. Estimation of non-normalized statistical models using score matching. *Journal of Machine Learning Research*, 6:695–709, 2005.
14. Pascal Lamblin and Yoshua Bengio. Important gains from supervised fine-tuning of deep architectures on large labeled sets. NIPS*2010 Deep Learning and Unsupervised Feature Learning Workshop, 2010.
15. Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.
16. Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the ACL-04 Workshop: Text Summarization Branches Out*, pages 74–81, 2004.
17. Rebecca J. Passonneau. Pyramid Annotation Guide: DUC 2006. www1.cs.columbia.edu/∼becky/DUC2006/2006-pyramid-guidelines.html, 2006.
18. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
19. David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
20. Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics(ACL2010)*, pages 384–394. Association for Computational Linguistics, July 2010.
21. Pascal Vincent. A connection between score matching and denoising autoencoders. Technical Report 1358, Université de Montréal, DIRO, November 2010.
22. Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *ICML '08: Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103, New York, NY, USA, 2008. ACM.
23. Ellen Vorhees and David Graff. AQUAINT-2 Information-Retrieval Text Research Collection. Linguistic Data Consortium, Philadelphia, 2008.
24. Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, pages 1168–1175. ACM, 2008.