Query-Based Single Document Summarization Using an Ensemble Noisy Auto-Encoder

Mahmood Yousefi Azar, Kairit Sirts, Diego Mollá Aliod and **Len Hamey**
Department of Computing
Macquarie University, Australia
mahmood.yousefiazar@students.mq.edu.au,
{kairit.sirts, diego.molla-aliod, len.hamey}@mq.edu.au

Abstract

In this paper we use a deep auto-encoder for extractive query-based summarization. We experiment with different input representations in order to overcome the problems stemming from sparse inputs characteristic to linguistic data. In particular, we propose constructing a local vocabulary for each document and adding a small random noise to the input. Also, we propose using inputs with added noise in an Ensemble Noisy Auto-Encoder (ENAE) that combines the top ranked sentences from multiple runs on the same input with different added noise. We test our model on a publicly available email dataset that is specifically designed for text summarization. We show that although an auto-encoder can be a quite effective summarizer, adding noise to the input and running a noisy ensemble can make improvements.

1 Introduction

Recently, deep neural networks have gained popularity in a wide variety of applications, in particular, they have been successfully applied to various natural language processing (NLP) tasks (Collobert et al., 2011; Srivastava and Salakhutdinov, 2012). In this paper we apply a deep neural network to query-based extractive summarization task. Our model uses a deep auto-encoder (AE) (Hinton and Salakhutdinov, 2006) to learn the latent representations for both the query and the sentences in the document and then uses a ranking function to choose certain number of sentences to compose the summary.

Typically, automatic text summarization systems use sparse input representations such as *tf-idf*. However, sparse inputs can be problematic in neural network training and they may make the training slow. We propose two techniques for reducing sparsity. First, we compose for each document a local vocabulary which is then used to construct the input representations for sentences in that document. Second, we add small random noise to the inputs. This technique is similar to the denoising auto-encoders (Vincent et al., 2008). However, the denoising AE adds noise to the inputs only during training, while during test time we also add noise to input.

An additional advantage of adding noise during testing is that we can use the same input with different added noise in an ensemble. Typically, an ensemble learner needs to learn several different models. However, the Ensemble Noisy Auto-Encoder (ENAE) proposed in this paper only needs to train one model and the ensemble is created from applying the model to the same input several times, each time with different added noise.

Text summarization can play an important role in different application domains. For instance, when performing a search in the mailbox according to a keyword, the user could be shown short summaries of the relevant emails. This is especially attractive when using a smart phone with a small screen. We also evaluate our model on a publicly available email dataset (Loza et al., 2014). In addition to summaries, this corpus has also been annotated with keyword phrases. In our experiments we use both the email subjects and annotated keywords as queries.

The contributions of the paper are the following:

1. We introduce an unsupervised approach for extractive summarization using AEs. Although AEs have been previously applied to summarization task as a word filter (Liu et al., 2012), to the best of our knowledge we are the first to use the representations learned by the AE directly for sentence ranking.
2. We add small Gaussian noise to the sparse input representations both during training and

Mahmood Yousefi Azar, Kairit Sirts, Len Hamey and Diego Mollá Aliod. 2015. Query-Based Single Document Summarization Using an Ensemble Noisy Auto-Encoder . In *Proceedings of Australasian Language Technology Association Workshop*, pages 2−10.

testing. To the best of our knowledge, noising the inputs during test time is novel in the application of AEs.

3. We introduce the Ensemble Noisy Auto-Encoder (ENAE) in which the model is trained once and used multiple times on the same input, each time with different added noise.

Our experiments show that although a deep AE can be a quite effective summarizer, adding stochastic noise to the input and running an ensemble on the same input with different added noise can make improvements.

We start by giving the background in section 2. The method is explained in section 3. Section 4 describes the input representations. The Ensemble Noisy Auto-Encoder is introduced in section 5. The experimental setup is detailed in section 6. Section 7 discusses the results and the last section 8 concludes the paper.

2 Background

Automatic summarization can be categorized into two distinct classes: abstractive and extractive. An abstractive summarizer re-generates the extracted content (Radev and McKeown, 1998; Harabagiu and Lacatusu, 2002; Liu et al., 2015). Extractive summarizer, on the other hand, chooses sentences from the original text to be included in the summary using a suitable ranking function (Luhn, 1958; Denil et al., 2014b). Extractive summarization has been more popular due to its relative simplicity compared to the abstractive summarization and this is also the approach taken in this paper.

Both extractive and abstractive summarizers can be designed to perform query-based summarization. A query-based summarizer aims to retrieve and summarize a document or a set of documents satisfying a request for information expressed by a user's query (Daumé III and Marcu, 2006; Tang et al., 2009; Zhong et al., 2015), which greatly facilitates obtaining the required information from large volumes of structured and unstructured data. Indeed, this is the task that the most popular search engines (e.g. Google) are performing when they present the search results, including snippets of text that are related to the query.

There has been some previous work on using deep neural networks for automatic text summarization. The most similar to our work is the model by Zhong et al. (2015) that also uses a deep AE for extractive summarization. However, they use the learned representations for filtering out relevant words for each document which are then used to construct a ranking function over sentences, while we use the learned representations directly in the ranking function. Denil et al. (2014a) propose a supervised model based on a convolutional neural network to extract relevant sentences from documents. Cao et al. (2015) use a recursive neural network for text summarization. However, also their model is supervised and uses hand-crafted word features as inputs while we use an AE for unsupervised learning.

The method of adding noise to the input proposed in this paper is very similar to the denoising auto-encoders (Vincent et al., 2008). In a denoising AE, the input is corrupted and the network tries to undo the effect of the corruption. The intuition is that this rectification can occur if the network learns to capture the dependencies between the inputs. The algorithm adds small noise to the input but the reconstructed output is still the same as uncorrupted input, while our model attempts to reconstruct the noisy input. While denoising AE only uses noisy inputs only in the training phase, we use the input representations with added noise both during training and also later when we use the trained model as a summarizer.

Previously, also re-sampling based methods have been proposed to solve the problem of sparsity for AE (Genest et al., 2011).

3 The Method Description

An AE (Figure 1) is a feed-forward network that learns to reconstruct the input $\boldsymbol{x}$. It first encodes the input $\boldsymbol{x}$ by using a set of recognition weights into a latent feature representation $C(\boldsymbol{x})$ and then decodes this representation back into an approximate input $\hat{\boldsymbol{x}}$ using a set of generative weights.

While most neural-network-based summarization methods are supervised, using an AE provides an unsupervised learning scheme. The general procedure goes as follows:

1. Train the AE on all sentences and queries in the corpus.
2. Use the trained AE to get the latent representations (a.k.a codes or features) for each query and each sentence in the document;
3. Rank the sentences using their latent representations to choose the query-related sentences to be included into the summary.

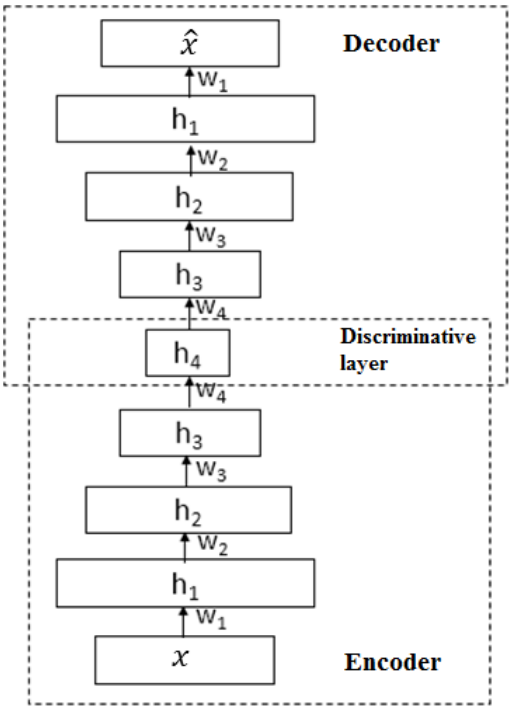

Figure 1: The structure of an AE for dimensionality reduction. x and $\hat{x}$ denote the input and reconstructed inputs respectively. h_i are the hidden layers and w_i are the weights. Features/codes $C(x)$ are in this scheme the output of the hidden layer h_4.

The AE is trained in two phases: *pre-training* and *fine-tuning*. Pre-training performs a greedy layer-wise unsupervised learning. The obtained weights are then used as initial weights in the fine-tuning phase, which will train all the network layers together using back-propagation. The next subsections will describe all procedures in more detail.

3.1 Pre-training Phase

In the pre-training phase, we used restricted Boltzmann machine (RBM) (Hinton et al., 2006). An RBM (Figure 2) is an undirected graphical model with two layers where the units in one layer are observed and in the other layer are hidden. It has symmetric weighted connections between hidden and visible units and no connections between the units of the same layer. In our model, the first layer RBM between the input and the first hidden representation is Gaussian-Bernoulli and the other RBMs are Bernoulli-Bernoulli.

The energy function of a Bernoulli-Bernoulli RBM, i.e. where both observed and hidden units are binary, is bilinear (Hopfield, 1982):

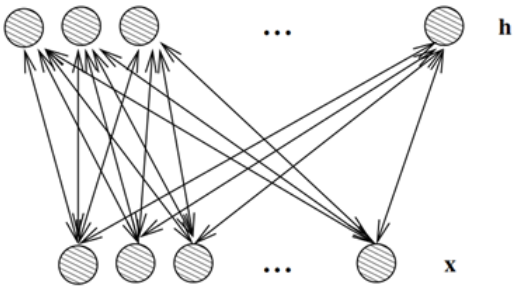

Figure 2: The structure of the restricted Boltzmann machine (RBM) as an undirected graphical model: x denotes the visible nodes and h are the hidden nodes.

$$E(\boldsymbol{x}, \boldsymbol{h}; \theta) = -\sum_{i \in V} b_i x_i - \sum_{j \in H} a_j h_j - \sum_{i,j} x_i h_j w_{ij}, \tag{1}$$

where V and H are the sets of visible and hidden units respectively, $\boldsymbol{x}$ and $\boldsymbol{h}$ are the input and hidden configurations respectively, w_{ij} is the weight between the visible unit x_i and the hidden unit h_j, and b_i and a_j are their biases. $\theta = \{\mathbf{W}, \mathbf{a}, \mathbf{b}\}$ denotes the set of all network parameters.

The joint distribution over both the observed and the hidden units has the following equation:

$$p(\boldsymbol{x}, \boldsymbol{h}; \theta) = \frac{\exp\left(-E(\boldsymbol{x}, \boldsymbol{h}; \theta)\right)}{Z}, \tag{2}$$

where $Z = \sum_{\boldsymbol{x}', \boldsymbol{h}'} \exp\left(-E(\boldsymbol{x}', \boldsymbol{h}'; \theta)\right)$ is the partition function that normalizes the distribution.

The marginal probability of a visible vector is:

$$p(\boldsymbol{x}; \theta) = \frac{\sum_{\boldsymbol{h}} \exp\left(-E(\boldsymbol{x}, \boldsymbol{h}; \theta)\right)}{Z} \tag{3}$$

The conditional probabilities for a Bernoulli-Bernoulli RBM are:

$$\begin{aligned} p(h_j = 1 | \boldsymbol{x}; \theta) &= \frac{\exp\left(\sum_i w_{ij} x_i + a_j\right)}{1 + \exp\left(\sum_i w_{ij} x_i + a_j\right)} \\ &= sigm(\sum_i w_{ij} x_i + a_j) \end{aligned} \tag{4}$$

$$\begin{aligned} p(x_i = 1 | \boldsymbol{h}; \theta) &= \frac{\exp\left(\sum_j w_{ij} h_j + b_i\right)}{1 + \exp\left(\sum_j w_{ij} h_j + b_i\right)} \\ &= sigm(\sum_j w_{ij} h_j + b_i) \end{aligned} \tag{5}$$

When the visible units have real values and the hidden units are binary, e.g. the RBM is Gaussian-Bernoulli, the energy function becomes:

$$E(\boldsymbol{x}, \boldsymbol{h}; \theta) = \sum_{i \in V} \frac{(x_i - b_i)^2}{2\sigma_i^2} - \sum_{j \in H} a_j h_j - \sum_{i,j} \frac{x_i}{\sigma_i} h_j w_{ij}, \quad (6)$$

where σ_i is the standard deviation of the ith visible unit. With unit-variance the conditional probabilities are:

$$p(h_j = 1 | \boldsymbol{x}; \theta) = \frac{\exp\left(\sum_i w_{ij} x_i + a_j\right)}{1 + \exp\left(\sum_i w_{ij} x_i + a_j\right)} = sigm(\sum_i w_{ij} x_i + a_j) \quad (7)$$

$$p(x_i | \boldsymbol{h}; \theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x - b_i - \sum_j w_{ij} h_j)^2}{2}\right) = \mathcal{N}\left(\sum_j w_{ij} h_j + b_i, 1\right) \quad (8)$$

To estimate the parameters of the network, maximum likelihood estimation (equivalent to minimizing the negative log-likelihood) can be applied. Taking the derivative of the negative log-probability of the inputs with respect to the weights leads to a learning algorithm where the update rule for the weights of a RBM is given by:

$$\Delta w_{ij} = \epsilon(\langle x_i, h_j \rangle_{data} - \langle x_i, h_j \rangle_{model}), \quad (9)$$

where ϵ is the learning rate, angle brackets denote the expectations and $\langle x_i, h_j \rangle_{data}$ is the so-called positive phase contribution and $\langle x_i, h_j \rangle_{model}$ is the so-called negative phase contribution. In particular, the positive phase is trying to decrease the energy of the observation and the negative phase increases the energy defined by the model. We use k-step contrastive divergence (Hinton, 2002) to approximate the expectation defined by the model. We only run one step of the Gibbs sampler, which provides low computational complexity and is enough to get a good approximation.

The RBM blocks can be stacked to form the topology of the desired AE. During pre-training the AE is trained greedily layer-wise using individual RBMs, where the output of one trained RBM is used as input for the next upper layer RBM (Figure 3).

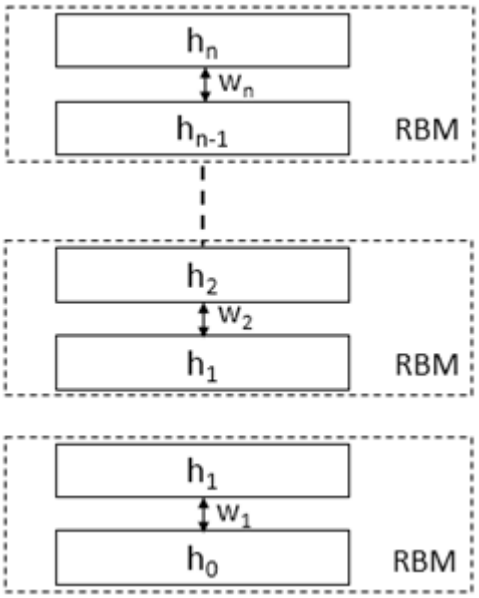

Figure 3: Several generative RBM models stacked on top of each other.

3.2 Fine-tuning Phase

In this phase, the weights obtained from the pre-training are used to initialise the deep AE. For that purpose, the individual RBMs are stacked on top of each other and unrolled, i.e. the recognition and generation weights are tied.

Ngiam et al. (2011) evaluated different types of optimization algorithm included stochastic gradient descent (SGD) and Conjugate gradient (CG). It has been observed that mini-batch CG with line search can simplify and speed up different types of AEs compared to SGD. In this phase, the weights of the entire network are fine-tuned with CG algorithm using back-propagation. The cost function to be minimised is the cross-entropy error between the given and reconstructed inputs.

3.3 Sentence Ranking

Extractive text summarization is also known as sentence ranking. Once the AE model has been trained, it can be used to extract the latent representations for each sentence in each document and for each query. We assume that the AE will place the sentences with similar semantic meaning close to each other in the latent space and thus, we can use those representations to rank the sentences accord-

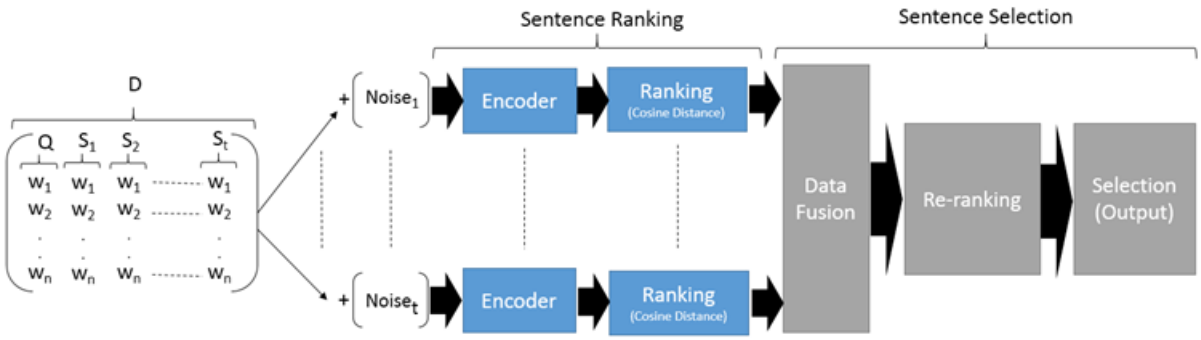

Figure 4: The Ensemble Noisy Auto-Encoder.

ing to their relevance to the query. We use cosine similarity to create the ranking ordering between sentences.

4 Input Representations

The most common input representation used in informations retrieval and text summarization systems is *tf-idf* (Wu et al., 2008), which represents each word in the document using its term frequency *tf* in the document, as well as over all documents (*idf*). In the context of text summarization the *tf-idf* representations are constructed for each sentence. This means that the input vectors are very sparse because each sentence only contains a small number of words.

To address the sparsity, we propose computing the *tf* representations using local vocabularies. We construct the vocabulary for each document separately from the most frequent terms occurring in that document. We use the same number of words in the vocabulary for each document.

This local representation is less sparse compared to the *tf-idf* because the dimensions in the input now correspond to words that all occur in the current document. Due to the local vocabularies the AE input dimensions now correspond to different words in different documents. As a consequence, the AE positions the sentences of different documents into different semantic subspaces. However, this behaviour causes no adverse effects because our system extracts each summary based on a single document only.

In order to reduce the sparsity even more, we add small Gaussian noise to the input. The idea is that when the noise is small, the information in the noisy inputs is essentially the same as in the input vectors without noise.

5 Ensemble Noisy Auto-Encoder

After ranking, a number of sentences must be selected to be included into the summary. A straightforward selection strategy adopted in most extractive summarization systems is just to use the top ranked sentences. However, we propose a more complex selection strategy that exploits the noisy input representations introduced in the previous section. By adding random noise to the input we can repeat the experiment several times using the same input but with different noise. Each of those experiments potentially produces a slightly different ranking, which can be aggregated into an ensemble.

In particular, after running the sentence ranking procedure multiple times, each time with different noise in the input, we use a voting scheme for aggregating the ranks. In this way we obtain the final ranking which is then used for the sentence selection. The voting scheme counts, how many times each sentence appears in all different rankings in the n top positions, where n is a predefined parameter. Currently, we use the simple counting and do not take into account the exact position of the sentence in each of the top rankings. Based on those counts we produce another ranking over only those sentences that appeared in the top rankings of the ensemble runs. Finally, we just select the top sentences according to the final ranking to produce the summary.

A detailed schematic of the full model is presented in Figure 4. The main difference between the proposed approach and the commonly used ensemble methods lies in the number of trained models. Whereas during ensemble learning several different models are trained, our proposed approach only needs to train a single model and the ensemble is created by applying it to a single input repeatedly, each time perturbing it with different noise.

6 Experimental Setup

We perform experiments on a general-purpose summarization and keyword extraction dataset (SKE) (Loza et al., 2014) that has been annotated with both extractive and abstractive summaries, and additionally also with keyword phrases. It consists of 349 emails from which 319 have been selected from the Enron email corpus and 30 emails were provided by the volunteers. The corpus contains both single emails and email threads that all have been manually annotated by two different annotators.

We conduct two different experiments on the SKE corpus. First, we generate summaries based on the subject of each email. As some emails in the corpus have empty subjects we could perform this experiment only on the subset of 289 emails that have non-empty subjects. Secondly, we generate summaries using the annotated keyword phrases as queries. As all emails in the corpus have been annotated with keyword phrases, this experiment was performed on the whole dataset. The annotated extractive summaries contain 5 sentences and thus we also generate 5 sentence summaries.

ROUGE (Lin, 2004) is the fully automatic metric commonly used to evaluate the text summarization results. In particular, ROUGE-2 recall has been shown to correlate most highly with human evaluator judgements (Dang and Owczarzak, 2008). We used 10-fold cross-validation, set the confidence interval to 95% and used the jackknifing procedure for multi-annotation evaluation (Lin, 2004).

Our deep AE implementation is based on G. Hinton's software, which is publicly available.[1] We used mini-batch gradient descent learning in both pre-training and fine-tuning phases. The batch size was 100 data items during pre-training and 1000 data items during fine-tuning phase. During pre-training we trained a 140-40-30-10 network with RBMs and in fine-tuning phase we trained a 140-40-30-10-30-40-140 network as the AE. Here, 140 is the size of the first hidden layer and 10 is the size of the sentence representation layer, which is used in the ranking function.

As a pre-processing step, we stem the documents with the Porter stemmer and remove the stop words.[2]

[1] `http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html`

[2] Stop word list obtained from `http://xpo6.com/list-of-english-stop-words`

Model	Subject	Phrases
tf-idf V=1000	0.2312	0.4845
tf-idf V=5%	0.1838	0.4217
tf-idf V=2%	0.1435	0.3166
tf-idf V=60	0.1068	0.2224
AE (*tf-idf* V=2%)	0.3580	0.4795
AE (*tf-idf* V=60)	0.3913	0.4220
L-AE	0.4948	0.5657
L-NAE	0.4664	0.5179
L-ENAE	0.5031	0.5370

Table 1: ROUGE-2 recall for both subject-oriented and key-phrase-oriented summarization. The upper section of the table shows *tf-idf* baselines with various vocabulary sizes. The middle section shows AE with *tf-idf* as input representations. The bottom section shows the AE with input representations constructed using local vocabularies (L-AE), L-AE with noisy inputs (L-NAE) and the Ensemble Noisy AE (L-ENAE).

We use *tf-idf*[3] as the baseline. After preprocessing, the SKE corpus contains 6423 unique terms and we constructed *tf-idf* vectors based on the 1000, 320 (5% of the whole vocabulary), 128 (2% of the whole vocabulary), and 60 most frequently occurring terms. $V = 60$ is the size of the *tf* representation used in our AE model. [4]

We apply the AE model to several different input representations: *tf-idf*, *tf* constructed using local vocabularies as explained in Section 4 (L-AE), *tf* using local vocabularies with added Gaussian noise (L-NAE) and in the noisy ensemble (L-ENAE).

7 Results and Discussion

Table 1 shows the ROUGE-2 scores of the *tf-idf* baselines and the AE model with various input representations. The columns show the scores of the summaries generated using the subjects and keyword phrases as queries respectively.

The main thing to note is that AE performs in most cases much better than the *tf-idf* baseline, especially when using subjects as queries. The only scenario where the *tf-idf* can compete with the AE

[3] We use the inverse frequency as the *idf* term.

[4] We did not train the AE-s with larger vocabularies because this would have required changing the network structure as during the preliminary experiments we noticed that the network did not improve much when using inputs larger than the first hidden layer.

is with the vocabulary of size 1000 and when using keyword phrases as queries. This is because the manually annotated keyword phrases do in many cases contain the same words as extracted summary sentences, especially because the queries and summaries where annotated by the same annotators. However, when the vocabulary size is the same as used in the AE, *tf-idf* scores are much lower in both experimental settings.

The second thing to notice is that although AE with *tf-idf* input performs better than plain *tf-idf*, it is still quite a bit worse than AE using *tf* representations derived from the local vocabularies. We believe it is so because the deep AE can extract additional information from the *tf-idf* representations, but the AE learning is more effective when using less sparse inputs, provided by the *tf* representations constructed from local vocabularies.

Although we hoped that the reduced sparsity stemming from the added noise will improve the results even more, the experiments show that this is not the case—AE without noise performs in both settings better than the noisy AE. However, when combining the rankings of the noisy ensemble, the results are much better than a single noisy AE and may even improve over the simple AE. This is the case when extracting summaries based on the subject. The subjects of the emails are less informative than the annotated keyword phrases. Perhaps this explains why the ENAE was able to make use of the noisy inputs to gain small improvements over the AE without any noise in this scenario.

There is considerable difference between the results when using the email subjects or keyword phrases as queries with keyword phrases leading to better summaries. This is to be expected because the keyword phrases have been carefully extracted by the annotators. The keyword phrases give the highest positive contribution to the *td-idf* baselines with largest vocabularies, which clearly benefits from the fact that the annotated sentences contain the extracted keyword phrases. The ENAE shows the smallest difference between the subject-based and keyword-based summaries. We believe it is because the ENAE is able to make better use of the whole latent semantic space of the document to extract the relevant sentences to the query, regardless of whether the query contains the exact relevant terms or not.

Figure 5 illustrates the ROUGE-2 recall of the best baseline and the AE models with both *tf-idf* and *tf* input representations using keyword phrases as queries and varying the length of the generated summaries. In this experiment, each summary was evaluated against the annotated summary of the same length. As is expected, the results improve when the length of the summary increases. While the AE model's results improve almost linearly over the 5 sentences, *tf-idf* gains less from increasing the summary length from 4 to 5 sentences. The scores are almost the same for the *tf-idf* and the AE with *tf* representation with 1 and 2 sentences. Starting from 3 sentences, the AE performs clearly better.

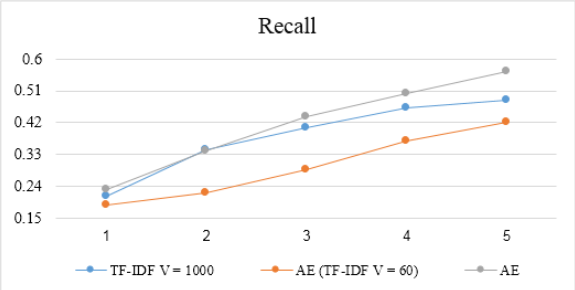

Figure 5: ROUGE-2 recall for summaries containing different number of sentences using the keyword phrases as queries.

To get a better feel what kind of summaries the ENAE system is generating we present the results of a sample email thread (ECT020). This typical email thread contains 4 emails and 13 lines. The summaries extracted by the ENAE system using both subjects and keyword phrases are given in Figure 6. The annotated summaries consist of sentences [03, 04, 10, 11, 05] and [03, 04, 11, 06, 05] for the first and the second annotator respectively.

Both generated summaries contain the sentences 03, 04 and 06. These were also the sentences chosen by the annotators (03 and 04 by the first annotation and all three of them by the second). The sentence 11 present in the subject-based summary was also chosen by both annotators, while sentence 10 in keyword-based summary was also annotated by the first annotator. The only sentences that were not chosen by the annotators are 08 in the subject-based summary and 12 in the keyword-based summary. Both annotators had also chosen sentence 05, which is not present in the automatically generated summaries. However, this is the sentence that both annotators gave the last priority in their rankings.

In general the order of the sentences generated by the system and chosen by the annotators is the

a) ENAE summary based on subject

03 Diamond-san, As I wrote in the past, Nissho Iwai's LNG related department has been transferred into a new joint venture company between Nissho and Sumitomo Corp. as of October 1, 2001, namely, "LNG Japan Corp.".

08 We are internally discussing when we start our official meeting.

04 In this connection, we would like to conclude another NDA with LNG Japan Corp, as per attached.

06 Also, please advise us how we should treat Nissho's NDA in these circumstances.

11 They need to change the counterparty name due to a joint venture.

b) ENAE summary based on keyword phrases

10 Please approve or make changes to their new NDA.

03 Diamond-san, As I wrote in the past, Nissho Iwai's LNG related department has been transferred into a new joint venture company between Nissho and Sumitomo Corp. as of October 1, 2001, namely, "LNG Japan Corp.".

04 In this connection, we would like to conclude another NDA with LNG Japan Corp, as per attached.

12 I wanted to let you know this was coming in as soon as Mark approves the changes.

06 Also, please advise us how we should treat Nissho's NDA in these circumstances.

Figure 6: Examples of subject-based (left) and keyword-based (right) summaries extrated by the Ensemble Noisy AE.

a) First annotator

LNG Japan Corp. is a new joint venture between Nissho and Sumitomo Corp. Given this situation a new NDA is needed and sent for signature to Daniel Diamond. Daniel forward the NDA to Mark for revision.

b) Second annotator

An Enron employee is informed by an employee of Nissho Iwai that the Nissho Iwai's LNG related department has been transferred into a new joint venture company, namely, 'LNG Japan Corp.'. As a result, there is a need to change the counterparty name in the new NDA. The new change has to be approved and then applied to the new NDA with LNG Japan Corporation

Figure 7: The abstractive summaries created by the annotators for the example email.

same in both example summaries. The only exception is sentence 10, which is ranked as top in the summary generated based on the keyword phrases but chosen as third after the sentences 03 and 04 by the first annotator.

Looking at the annotated abstractive summaries we found that the sentence 12 chosen by the keyword-based summarizer is not a fault extraction. Although neither of the annotators chose this sentence for the extractive summary, the information conveyed in this sentence can be found in both annotated abstractive summaries (Figure 7).

8 Conclusion

In this paper we used a deep auto-encoder (AE) for query-based extractive summarization. We tested our method on a publicly available email dataset and showed that the auto-encoder-based models perform much better than the *tf-idf* baseline. We proposed using local vocabularies to construct input representations and showed that this improves over the commonly used *tf-idf*, even when the latter is used as input to an AE. We proposed adding small stochastic noise to the input representations to reduce sparsity and showed that constructing an ensemble by running the AE on the same input multiple times, each time with different noise, can improve the results over the deterministic AE.

In future, we plan to compare the proposed system with the denoising auto-encoder, as well as experiment with different network structures and vocabulary sizes. Also, we intend to test our Ensemble Noisy Auto-Encoder on various different datasets to explore the accuracy and stability of the method more thoroughly.

References

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summa-

rization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Ronan Collobert, Jason Weston, Lon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the tac 2008 update summarization task. In *Proceedings of text analysis conference*, pages 1–16.

Hal Daumé III and Daniel Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of ACL06*, pages 305–312. Association for Computational Linguistics.

Misha Denil, Alban Demiraj, and Nando de Freitas. 2014a. Extraction of salient sentences from labelled documents. *arXiv preprint arXiv:1412.6815*.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014b. Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv preprint arXiv:1406.3830*.

Pierre-Etienne Genest, Fabrizio Gotti, and Yoshua Bengio. 2011. Deep learning for automatic summary scoring. In *Proceedings of the Workshop on Automatic Text Summarization*, pages 17–28.

Sanda M Harabagiu and Finley Lacatusu. 2002. Generating single and multi-document summaries with gistexter. In *Document Understanding Conferences*.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

John J Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

Yan Liu, Sheng-hua Zhong, and Wenjie Li. 2012. Query-oriented multi-document summarization via unsupervised deep learning. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations.

Vanessa Loza, Shibamouli Lahiri, Rada Mihalcea, and Po-Hsiang Lai. 2014. Building a dataset for summarization and keyword extraction from emails. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research Development*, 2(2):159.

Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc V Le, and Andrew Y Ng. 2011. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 265–272.

Dragomir R Radev and Kathleen R McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):470–500.

Nitish Srivastava and Ruslan R Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230.

Jie Tang, Limin Yao, and Dewei Chen. 2009. Multi-topic based query-oriented summarization. In *SDM*, volume 9, pages 1147–1158. SIAM.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):13.

Sheng-hua Zhong, Yan Liu, Bin Li, and Jing Long. 2015. Query-oriented unsupervised multi-document summarization via deep learning model. *Expert Systems with Applications*, 42(21):8146–8155.