applied sciences

MDPI

Article

Semantic Features Based N-Best Rescoring Methods for Automatic Speech Recognition

Chang Liu [1,2], Pengyuan Zhang [1,2], Ta Li [1,2,*] and Yonghong Yan [1,2,3]

1 Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China; liuchang@hccl.ioa.ac.cn (C.L.); zhangpengyuan@hccl.ioa.ac.cn (P.Z.); yanyonghong@hccl.ioa.ac.cn (Y.Y.)

2 University of Chinese Academy of Sciences, Beijing 100049, China

3 Xinjiang Key Laboratory of Minority Speech and Language Information Processing, Xinjiang Technical Institute of Physics and Chemistry, Chinese Academy of Sciences, Urumqi 830011, China

* Correspondence: lita@hccl.ioa.ac.cn

Received: 16 October 2019; Accepted: 18 November 2019; Published: 22 November 2019

Abstract: In this work, we aim to re-rank the n-best hypotheses of an automatic speech recognition system by punishing the sentences which have words that are semantically different from the context and rewarding the sentences where all words are in semantical harmony. To achieve this, we proposed a topic similarity score that measures the difference between topic distribution of words and the corresponding sentence. We also proposed another word-discourse score that quantifies the likeliness for a word to appear in the sentence by the inner production of word vector and discourse vector. Besides, we used the latent semantic marginal and a variation of log bi-linear model to get the sentence coordination score. In addition we introduce a fallibility weight, which assists the computation of the sentence semantically coordination score by instructing the model to pay more attention to the words that appear less in the hypotheses list and we show how to use the scores and the fallibility weight in hypotheses rescoring. None of the rescoring methods need extra parameters other than the semantic models. Experiments conducted on the Wall Street Journal corpus show that, by using the proposed word-discourse score on 50-dimension word embedding, we can achieve 0.29% and 0.51% absolute word error rate (WER) reductions on the two testsets.

Keywords: automatic speech recognition (ASR); semantic model; topic model; continuous word representation

1. Introduction

Language modeling, which learns the probability distribution of a given word and captures the extent to which a sequence of words can be considered well formed, is an important part of automatic speech recognition (ASR) tasks. It provides a context to distinguish between words and phrases that are pronounced in a similar way. This is because n-gram language models use the assumption that the probability of the i-th word w_i given the context history of the preceding $i-1$ words can be approximated by the probability of observing it in the context history of the preceding $n-1$ words. Generally, n-gram language models are used in the first-pass decoding process in ASR tasks as only a small number of historical words are needed. Along with the 1-best hypotheses, which could be the final outputs of the system, lattices and n-bests hypotheses can be generated by the first pass decoding process. Other language models like neural network language models are usually used to rescore lattices or n-best hypotheses [1,2].

Here is an example of some of the hypotheses of the utterance named "446c040q" of a Wall Street Journal (WSJ) dataset:

- 446c040q-1 THERE ARE INDICATIONS THAT SALES ARE SLOWING DOWN BUT CONSUMER CREDIT SEARCH UPWARD IN DECEMBER
- 446c040q-2 THERE ARE INDICATIONS THAT SALES ARE SLOWING DOWN BUT CONSUMER CREDIT SURGED UPWARD IN DECEMBER
- 446c040q-3 THERE ARE INDICATIONS THAT SALES ARE SLOWING DOWN BUT CONSUMER CREDIT SIR <UNK> UPWARD IN DECEMBER

Even without listening to the original speech, people who can speak English and have common knowledge can easily point out that the second hypothesis is the most possible one as the word "surge" fits the topic of the sentence more and only with "surge" does the sentence make sense.

Inspired by this, we propose to use a score to measure whether a word is abrupt in the context according to semantic information, and use the score to rescore the hypotheses to promote the performance of ASR systems.

Most of the work of using semantic features is based on neural network language models. Neural network language models can often offer excellent results, but they have a severe requirement of machine and time for training, especially when the training set or vocabulary set is very large. In addition, neural language models lack flexibility. A small modification on the vocabulary list could lead to retraining the whole network. Thus, in this work, we offer a different option by using the semantic features without neural networks.

Our task has some similarity with ASR error detection. ASR error detection focuses on distinguishing whether the words in ASR results are wrong by using both decoder based and non-decoder based features. The methods are usually tested on the classification error rate of classifying words to right and wrong classes. Generally speaking, the 1-best hypothesis (hypothesis with the highest score of the corresponding utterance) is not always the hypothesis which leads to the lowest WERs. Therefore, an effective hypotheses selection can be of great help. Thus, in this work we do not focus on differentiating whether a certain word is right or wrong, but propose a score to evaluate on what degree the word is appropriate in the hypothesis sentence by semantic features. In this work, we use the latent semantic marginal (LSM) [3] and a variation of log bi-linear language model [4] to form a sentence coordination score. In addition to that, we propose a word-sentence topic similarity score and a word-discourse probability score based on the theory of Arora's text generation hypotheses [5]. All the four scores are designed to be light-weight, i.e., they do not need parameters other than topic models or word embedding models. We use the four scores in the n-best rescoring process and show the effectiveness of each score on improving ASR systems.

As shown by the previous example, there are many common words in hypotheses lists. Moreover, it is obvious that if a word appears at similar positions in all the hypotheses, whether this word is the correct word does not change the result of the rescoring process. We should focus more on the words that have more candidates. Thus, we first align the hypotheses to group the words in different hypotheses which are the recognized results for the same time period. Then we introduce a fallibility weight, which assists the computation of the sentence's semantic coordination scores by instructing the model to pay more attention to the words that may have more possible choice.

The structure of this paper is as follows: In Section 2, we introduce some related works. In Section 3, we introduce the topic model and word embeddings which are the semantic features we use in this research. In Section 4, we present our rescoring method. We first introduce the four sentence scores. Then we describe the fallibility weight of words. Finally we introduce how we use the sentence scores in the ASR systems. We present experiments in Section 5 to validate our method and discuss some of the parameters in the method. Section 6 presents the conclusion.

2. Related Works

Topic information as a kind of semantic information has been used in language models in many works. Chu and Mangu performed a hard partitioning of the data according to the topics and built a set of disjoint models to depict data of different topics respectively [6]. Jin et al. [7] mapped each paragraph into a unique vector in continuous space and performed unsupervised clustering to construct topic clusters and then built several topic-specific language models like [6]. Lau et al. [8] proposed a neural language model that incorporates document context in the form of a topic model-like architecture to provide a succinct representation of the broader document context outside of the current sentence. Latent Dirichlet allocation (LDA) [9] is the most popular topic modelling algorithm because it is less prone to overfitting and it can be used to compute the probability of an unobserved document [3]. Mikolov et al. proposed using topic information from LDA as an external feature in recurrent neural network language models [10]. LDA has been used in language model adaption in [3,11–13]. It has also been used to divide a training set into clusters by the topic distribution in [14,15].

There is an hypotheses generator to forecast air traffic control voice commands taking into account context knowledge [16]. However, this concentrates more on a semantic error rate than on a word error rate. The hypotheses generation for air traffic control can also be improved by machine learning [17].

Word embedding [18–20] projects each word to a vector in a continuous vector space, by making words with similar context closer in the space. Many works have analyzed the result or the word vectors by experiments on polysemy or studies on the similarity of nearby words or sentences [5,21,22]. Most works using word embedding in rescoring n-best also build on the work of neural network language models. Audhkhasi et al. [23] used word embedding as inputs in his proposed feedforward neural network language models' architecture. Besides adding word embedding to the hidden layer, character embeddings are additionally added to the hidden layer and output later in a Chinese speech recognition task in by He and Xiang et al. [24].

3. Topic Model and Word Embeddings

In this section, we introduce the topic model and word semantic vector model that are used in the method we propose.

3.1. Latent Dirichlet Allocation for Topic Modelling

Latent Dirichlet Allocation (LDA) [9] is a text generation model which has a hierarchy structure of document, topic and word. For a document, the generation process can be described as follows:

- Sample the length N of the document from Poisson distribution: $N \sim Poission(\xi)$.
- Sample a multinomial distribution over topics for document i from a Dirichlet distribution parameterized by α: $\Theta_i \sim Dir(\alpha)$.
- For the j-th word in the document, sample the topic of this word, $z_{i,j}$, from Θ_i, $z_{i,j} \sim Multinomial(\Theta_i)$ and then sample the word $w_{i,j}$ from the unigram distribution given the topic: $w_{i,j} \sim p(w_{i,j}|z_{i,j}, \beta)$.

The key parameter α controls the sharpness of probability distribution over topics when choosing topics for a document. When $\alpha = 1$, the topic is uniformly picked. When $\alpha > 1$, the probabilities of more uniformed ones are given with larger probabilities. When α is < 1, the probability tends to peaked distributions over the topic. Our experiment uses the LDA implementation by Blei et al. (https://github.com/blei-lab/lda-c).

When given the training data, topic number and the initial α, the algorithm learns the best value of α, the word generation probability of topic β and the topic distribution γ for the training documents by iteration. There is also an inference program in the LDA tool kit to tell the topic distribution γ for the given document from the model learned by the training procedure.

3.2. Word Embedding

Word embeddings are distributed representations of words in a vector space, which help learning algorithms to achieve better performance in natural language processing tasks by grouping similar words. Many different approaches have been investigated to generate the word embeddings of word given some training text. In this paper, we use the widely used GloVe algorithm [20] to generate word embeddings. GloVe is based on the co-occurrences of words in a window. Let X be the word-word co-occurrence matrix, of which entry $x_{i,j}$ denote the count of i-th and j-th word co-occurrence in a window. Let $x_i = \sum_j x_{i,j}$ denote the number of times any word appears in the context of the i-th word.

The GloVe can be regarded as a weighted least square regression with the cost function as follows:

$$J = \sum_{i,j=1}^{V} f(X_{i,j})(w_i^{\mathrm{T}} \tilde{w}_j + b_i + \tilde{b}_j - log X_{i,j})^2, \tag{1}$$

where V is the size of vocabulary, $w \in \mathbb{R}^d$ and $\tilde{w} \in \mathbb{R}^d$ are word vectors, b and $\tilde{b}$ are biases. As X is symmetric, W and $\tilde{W}$ are equivalent and differ only as a result of their random initializations. The sum $W + \tilde{W}$ is the final output of the algorithm.

The weighted function should have the following properties:

1. $f(0) = 0$. If f is viewed as a continuous function, it should obey that $\lim_{x \to 0} f(x) log^2 x$ is finite.
2. $f(x)$ should be non-decreasing in case that rare co-occurrences are overweighted.
3. $f(x)$ should be relatively small for large value of x, in case that frequent co-occurrences are overweighted.

In practice, $f(x)$ are usually chosen as follows:

$$f(x) = \begin{cases} (x/x_{max})^{\alpha}, & x > x_{max} \\ 1, & otherwise \end{cases} \tag{2}$$

Figure 1 shows the model architecture of GloVe. The input is a one-hot representation of a word. The word embedding matrices serve as weight matrices in the model and thus the output of the model is a vector of inner products of word vectors. The embedding matrices are updated by the gradient of the loss function introduced before in this section.

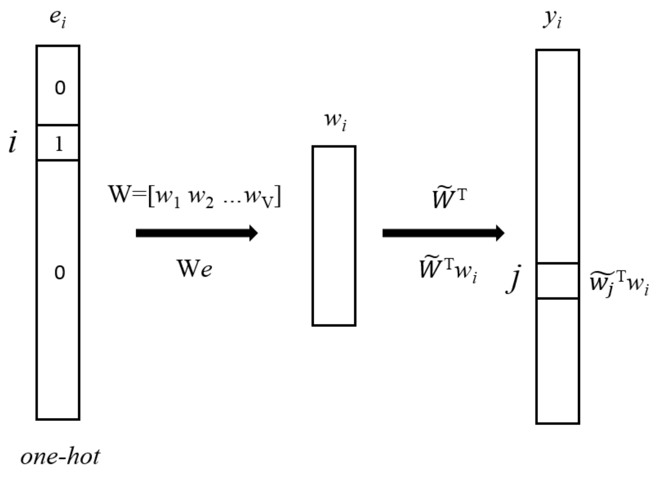

e_i
0
i
1
0
one-hot
W=[w_1 w_2 ...w_V]
We
w_i
$\widetilde{W}^{\mathrm{T}}$
$\widetilde{W}^{\mathrm{T}} w_i$
y_i
j
$\widetilde{w_j}^{\mathrm{T}} w_i$

Figure 1. The model architecture of GloVe. The input is a one-hot representation of a word. The word embedding matrices serve as weight matrices in the model and thus the output of the model is a vector of inner products of word vectors.

The text generation process is modelled as a random walk in the word vector space by Hashimoto et al. [22]. Given that the last word produced was w, the probability that the next word is w' is assumed to be given by $h(|v_w - v_{w'}|^2)$ for a suitable h. Arora et al. [5,21] further proposed a latent discourse vector in the random walk model, which has a clear semantic interpretation and has been proven useful in subsequent work, e.g., in their work of understanding structure of word embeddings for polysemous words.

In their model, the text generation process is driven by the discourse vector $c_t \in \mathbb{R}^d$, which is a vector to express what is being talked about. Each word has a continuous representation $w_t \in \mathbb{R}^d$ and the probability of a word is emitted at time t given the discourse vector is modelled with a log-linear word production model by Arora [21]

$$P[w \text{ emitted at time } t|c_t] \propto exp(c_t^T w_t). \tag{3}$$

The discourse vector c_t does a slow random walk, and c_{t+1} can be obtained by adding a small random vector from c_t. Therefore, nearby words are generated under similar discourse.

With the assumption that the discourse vector c_t does not change much when modelling the word emitting procedure, it can be simplified to replace all the c_t's in the sentence s with a sentence discourse c_s. It was shown in the paper of Arora et al. [5,21] that the maximum a posteriori estimation of c_s is the average of word embeddings for all the words in the sentence. So, the word emitting probability can be writen as follows:

$$P[w \text{ emitted at time } t|c_s] \propto exp(c_s^T w_t), \tag{4}$$

$$c_s = \frac{1}{|S|}\sum_{i=1}^{|S|} w_{s_i}. \tag{5}$$

4. N-Best Rescoring with Coordination Scores

In this section, we first introduce the scores, which are used for evaluating how much is the sentence coordinate in topic or semantic. And then we introduce how we use the coordination scores to rerank the n-best hypothesis.

4.1. Sentence Coordination Scores

4.1.1. Sentence Probability Score Using LDA

To discriminate whether a word is suitable in a document, we could use the probability of a word given the topic estimation of the document. The probability can be computed by a weighted average of the probability of this word given each topic, for which the weight is the topic distribution of the document, i.e.,

$$P(w_i|s) = \sum_{j=1}^{|T|} P(w_i|z_j)P(z_j|s), \tag{6}$$

where $|T|$ is the hyper-parameter of topic numbers in the LDA model training progress, and z_j represent the j-th topic. We use the item in Row i Column j of the β matrix for $P(w_i|z_j)$ calculated by the LDA tool kit mentioned in the previous section and the corresponding row in γ for the hypothesis sentence s by the infer procedure in the tool kit for $P(z_j|s)$. In re-ranking tasks, we regard the hypothesis sentences as the given document.

This probability is regarded as LSM [3] and it is used to measure the coordination of a sentence for the first time here.

The probability of a word sequence, which is regarded as a bag-of-word (BOW) in this word, given the sentence topic distribution of a sentence is the product of the probabilities for each words in the sentence given the sentence topic distribution when simplified that the words are independent. It can be written as:

$$P(w_1,\ldots,w_{|L|}|s) = \prod_{i=1}^{L} P(w_i|s) = \prod_{i=1}^{L}[\sum_{j=1}^{|T|} P(w_i|z_j)P(z_j|s)]. \tag{7}$$

where L is the length of the document, namely the hypothesis sentence. We use the logarithm of the word sequence probability as the sentence coordination score:

$$Score_s = logP(w_1, \dots, w_{|S|}|s). \tag{8}$$

4.1.2. Topic Similarity Score Using LDA

To discriminate the mistakenly recognized word that differs in the topic with the sentence s, we use the cosine similarity to measure word-sentence topic bias:

$$c_{w_i} = \frac{< P(z|w_i), P(z|s) >}{|P(z|w_i)||P(z|s)|} = \frac{\sum_{k=1}^{|T|} P(z_k|w_i)P(z_k|s)}{\sqrt{\sum_{k=1}^{|T|} P(z_k|w_i)^2}\sqrt{\sum_{k=1}^{|T|} P(z_k|s)^2}} \tag{9}$$

where $P(z|w_i)$ and $P(z|s)$ are both $|T|$-dimensional vectors, which are the topic distributions of w_i and sentence s respectively. Provided that in LDA algorithm, all the topics are equally modelled in the training process ($P(z_j) = 1/|T|$, for all j), the probability of topic distribution given word can be normalized by the word probability distribution of each topic:

$$P(z_j|w_i) = \frac{P(w_i|z_j)P(z_j)}{\sum_{k=1}^{|T|} P(w_i|z_k)P(z_k)} \tag{10}$$

$$= \frac{P(w_i|z_j)}{\sum_{k=1}^{|T|} P(w_i|z_k)}, \tag{11}$$

Thus the topic probability given word can be calculated by column normalization of the β matrix from the LDA pre-training model.

The range of cosine similarity is $[-1, 1]$, where the value close to 1 means that the topic of the word is very similar to the sentence, and in contrast the value near -1 means that the topic of the word is nearly the opposite to the sentence.

We calculate the sentence topic similarity score by simply adding the word similarities:

$$Score_s = \sum_{i=1}^{|S|} c_{w_i} \tag{12}$$

4.1.3. Word-Pair Probability Score Using Word Embedding

Let $p_{i,j}$ be the probability that word j appear in the context of word i. It is apparent that $p_{i,j} = x_{i,j}/x_i$. And according to the last section, the logarithm of $x_{i,j}$ can be approximate to $w_i^T w_j$. Thus, $p_{i,j}$ can be calculated as follows:

$$p_{i,j} = \frac{x_{i,j}}{x_i} = \frac{exp(w_i^T w_j)}{\sum_{k=1}^{V} exp(w_i^T w_k)} = softmax(W^T w_i)_j, \tag{13}$$

where $W = [w_1, w_2, \dots, w_V]$ and $(\bullet)_j$ refer to the j-th element of $(\bullet)$. The calculation can be further modified by γ to control the shape of the probability distribution as follows:

$$p_{i,j} = \frac{x_{i,j}}{x_i} = \frac{exp(\gamma w_i^T w_j)}{\sum_{k=1}^{V} exp(\gamma w_i^T w_k)} = softmax(\gamma W^T w_i)_j, \tag{14}$$

γ is the smoothing factor, which can be regarded as a stretch of all word vectors. γ controls the shape of the probability distribution. When $\gamma > 1$, the probability distribution will be sharpen and when $\gamma < 1$, the distribution will be flatten.

The word pair probability is similar to the log bi-linear (LBL) language model [4]. It can be regarded as a bi-gram language model which is smoothed by the inner product of the continuous expression for the rare word-pairs. To avoid repetitive computation and accelerate the calculation, we can pre-compute and store all x_i. By pre-computing, the probability of a word pair can be calculated in $O(d)$ time. d is the length of word vector.

Usually, the context of a word is defined by the previous two words and the preceding two words of the current word. We use this setting for both word embedding training and word-pair probability prediction. We use the average of conditional probabilities of the current word given words in the context of the current word. The probability can be expressed as follows:

$$p_i = \frac{p_{i-2,i} + p_{i-1,i} + p_{i+1,i} + p_{i+2,i}}{k} \tag{15}$$

when $i-2$ or $i-1$ is less than 0, or $i+1$, $i+2$ is larger than the length of the sentence, we use zero as the word-pair probability. And k is the number of non-zero probabilities of the 4 items. The word probability score for the i-th word is the logarithm of p_i.

We used a window with exponential weights to emphasize the words near the current word and weaken the effect of words further to the current word. But the result is at a similar level to that of the noncentral rectangular window.

4.1.4. Word-Discourse Probability Score Using Word Embedding

As explained in the second section, the probability of a word given a sentence is proportional to the exponent of the inner product of the word embedding of the word and the discourse vector, which can be simplified as the average of word embeddings of all the words in the sentence. The probability of the t-th word in sentence s is calculated as follows:

$$p(w_t|s) = \frac{exp(v_t^T c_s)}{\sum_{k=1}^{V} exp(w_k^T c_s)} = softmax(W^T c_s)_{w_t}, \tag{16}$$

where w_t is the word embedding of the t-th word in sentence s, V is the size of vocabulary. c_s is the discourse vector introduced in the previous section. As all words in the same sentence share the same discourse vector, we firstly compute c_s when we begin to process the sentence. Therefore, the computation of word-discourse probability score is more time-consuming than the word-pair probability score. Similar to the topic similarity score using LDA, the sentence word-discourse probability score is the sum of the logarithms of word-discourse probabilities for all the words in the sentence.

4.2. Fallibility of Word for ASR Hypothesis

The n-best hypothesis of ASR result has a trait that given any two hypothesis for the same utterance, the mass of the hypothesis pairs have only a few words that are different, which can be instantiated by the example in the introduction.

To rerank the hypothesis of the ASR result, we tend to focus on the different words in the hypotheses and ignore the words that exists in every hypothesis. Motivated by this, we introduce the notion of fallibility of word to the rescoring progress to magnify the influence of the words that exists less in similar places in the hypothesis list.

We first align all the hypothesis pairs for each utterance. For each word in a hypothesis, we collect all the words that correspond to the required word in the aligning results in a set. (Each time the required word aligned with blank, we regard blank as a unique word.) The fallibility of a word in a hypothesis is defined by the number of unique words (namely number of types) for the words that are different from the required word in the set. For example, the fallibility weight for the word "surged" in hypothesis 446c040q-2 (presented in the introduction) is 2 as word "search" and "sir" both correspond to and different from "surged". Figure 2 is an example to show how to compute the fallibility.

Hy1	A	B	C	E	D
Hy2	A	F	C	×	D
Hy3	A	B	C	×	G
Hy4	A	×	C	×	D
Different types	1	3	1	2	2
Fallibility for words in hy1	0	2	0	1	1

Figure 2. An example of the fallibility.

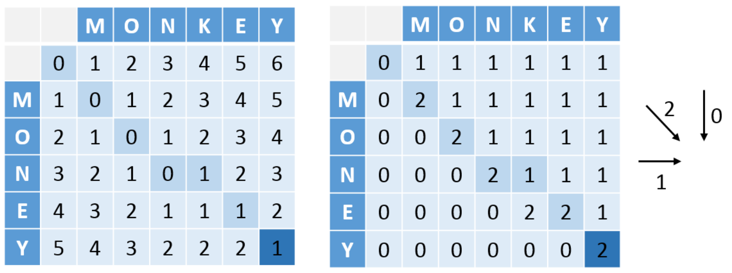
The aligning operation can be implemented by the minimum edit distance algorithm [25]. We use $F(m,n)$ to denote the minimum edit distance for the first m words and first n words for the two sentences, S_1 and S_2. $F(m,n)$ is computed by $F(m-1,n)$, $F(m,n-1)$ or $F(m-1,n-1)$, each of which represent a way of tagging the present word $S_1(m)$ and word $S_2(n)$:

$$F(m,n) = min \begin{cases} F(m-1,n)+1 \\ F(m,n-1)+1 \\ F(m-1,n-1)+\begin{cases} 0, if S_1(m) = S_2(n) \\ 1, if S_1(m) \neq S_2(n) \end{cases} \end{cases}, F(0,n) = n, F(m,0) = m. \quad (17)$$

$F(|S1|,|S2|)$ is the final minimum edit distance of the two sentences. We can keep the path by which F item $F(m,n)$ is computed and tag each word to distinguish the word to be correct or error.

Figure 3 is an example of the minimum edit distance algorithm. In this example we align the word "monkey" and the word "money" by character. The left table is the F matrix and the bottom right element is the final edit distance for the two words. The right matrix stores the path. For the (i,j)-th element, "0", "1" and "2" mean that $F(i,j)$ is assigned by $F(m-1,n)+1$, $F(m,n-1)+1$ and $F(m-1,n-1)+0/1$ respectively. From the path matrix, we can label words in hypotheses with "correct", "substitute" and "insert" tags. For example, if the vertical words refer to the reference and the horizontal words refer to hypothesis, "1" means that the corresponding word in hypothesis is of "insert" tag, and "2" means that the word is of "correct" tag or "substitute" tag, which depends on whether the corresponding words are different. We could not give a word "delete" tag in hypothesis, because the "deleted" word is not in the hypothesis.

To emphasize the fallible words, we multiply the word score with the word fallibility score when computing the sentence score.



		M	O	N	K	E	Y
	0	1	2	3	4	5	6
M	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
N	3	2	1	0	1	2	3
E	4	3	2	1	1	1	2
Y	5	4	3	2	2	2	1

		M	O	N	K	E	Y
	0	1	1	1	1	1	1
M	0	2	1	1	1	1	1
O	0	0	2	1	1	1	1
N	0	0	0	2	1	1	1
E	0	0	0	0	2	2	1
Y	0	0	0	0	0	0	2

Figure 3. An example of the minimum edit distance algorithm. The left sub-figure show the F matrix, which is the minimum edit distance for the sub-sentences. The right sub-figure shows the path matrix, from which we can get the alignment.

4.3. n-Best Rescoring for ASR

Hypotheses of ASR are ranked by the sum of acoustic model score ($Score_{AM}$) and language model score ($Score_{LM}$), which are the logarithms of the state probability and word sequence probability respectively:

$$Score(s) = Score_{AM}(s) + \lambda Score_{LM}(s) \tag{18}$$

We interpolate the score of the base language model and the similarity or probability score proposed in last subsection by α to get the new language model score:

$$Score(s) = Score_{AM}(s) + \lambda[\alpha Score_{LM}(s) + (1 - \alpha) \cdot k \cdot Score_{proposed}(s)] \tag{19}$$

where k is a normalizer as the scale of $Score_{proposed}$ might differ greatly with that of $Score_{LM}$ especially for the scores with fallibility weights. We use the proportion between the median of $Score_{proposed}$ and $Score_{LM}$ for k at first and change the value of it to get the best result.

5. Experiments

5.1. Experimental Setup

In this study the proposed rescoring method is carried out on the WSJ speech recognition task (https://catalog.ldc.upenn.edu/docs/LDC94S13A/wsj1.txt). The dataset contains an 80-hour training set and a text corpus which has 37M tokens, and 162K words for language modeling. We use Kaldi [26] for acoustic model training and decoding. The acoustic model is a time-delay neural network (TDNN) with 6 layers, each layer with 250 hidden units. The acoustic model trained by the cross-entropy (CE) loss. The first-pass decode uses a 3-gram language model with Kneser-Ney (K-N) smoothing by srilm [27]. The large text corpus is used for n-gram and the small RNN language models and the training of the semantic models for LDA and GloVe. In this study, dev93 and eval92 are used equally to evaluate the proposed methods.

The number of hypotheses used for re-ranking is 100 for both eval92 and dev93. We follow the kaldi's recipe to enumerate from 9 to 20 to find the best λ in Equation (19). The best choice for eval92 is 13. Both 12 and 13 yield the best results for dev93. We use 13 for both testsets in rescoring.

5.2. LDA-Based Rescoring

In this section, we use LDA-based scores to re-rank hypotheses.

The average scores of hypotheses at different ranks are in Figure 4. The upper sub-figures show the average logarithms of sentence probability and the ones at the bottom are the averages of topic similarity. "LDAprob" refer to the LDA sentence probability score, and "LDAtopsim" refer to the topic similarity score in this table. The tendency of the curves shows that the hypotheses ranked at the front are of higher scores than those ranked at the back for all sub-figures. Given the assumption that generally hypotheses ranked in front are of better quality, the curves could reflect this to some extent. The average scores with fallibility are in the right two sub-figures.

We aligned all the hypotheses in both testsets with references and tag words in hypotheses with 'correct', 'substitute' and 'insert' by the aligning algorithm described in Section 3.2. Deletion error cannot be reported in hypotheses, so we do not have a 'delete' tag. Figure 5 shows the distribution of the two word scores for words with 'correct', 'substitute' and 'insert' tags respectively. The sub-figures on the top are those of word probability score, and the bottom ones are of topic similarity score. The statistic of the averaged score for words with 'correct' 'substitute' and 'insert' tags is listed in Table 1. As shown in the table, for both score-types, the average score of the 'correct' tags are the highest and the average of 'insert' words are much lower than 'correct' and 'substitute'. This shows that the two scores can discriminate the wrong words in the hypotheses to a certain extent.

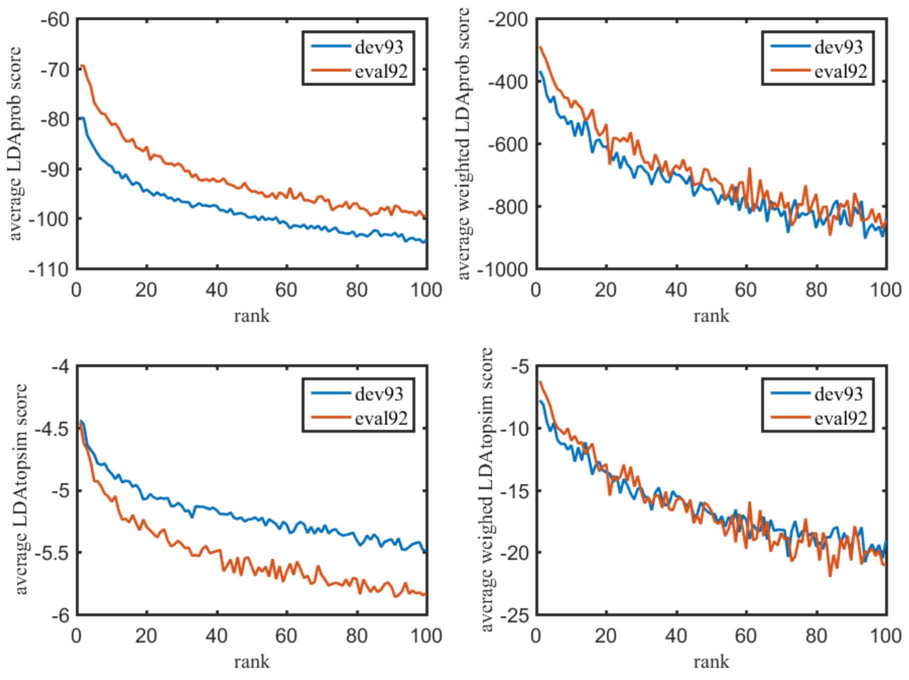


Figure 4. The average score of Latent Dirichlet allocation (LDA)-based methods for hypotheses. The horizontal axis, ranging from 1 to 100, represents the rank of hypotheses by the 1-pass decoding process. The vertical axis represents the average score of the hypotheses at the corresponding rank.

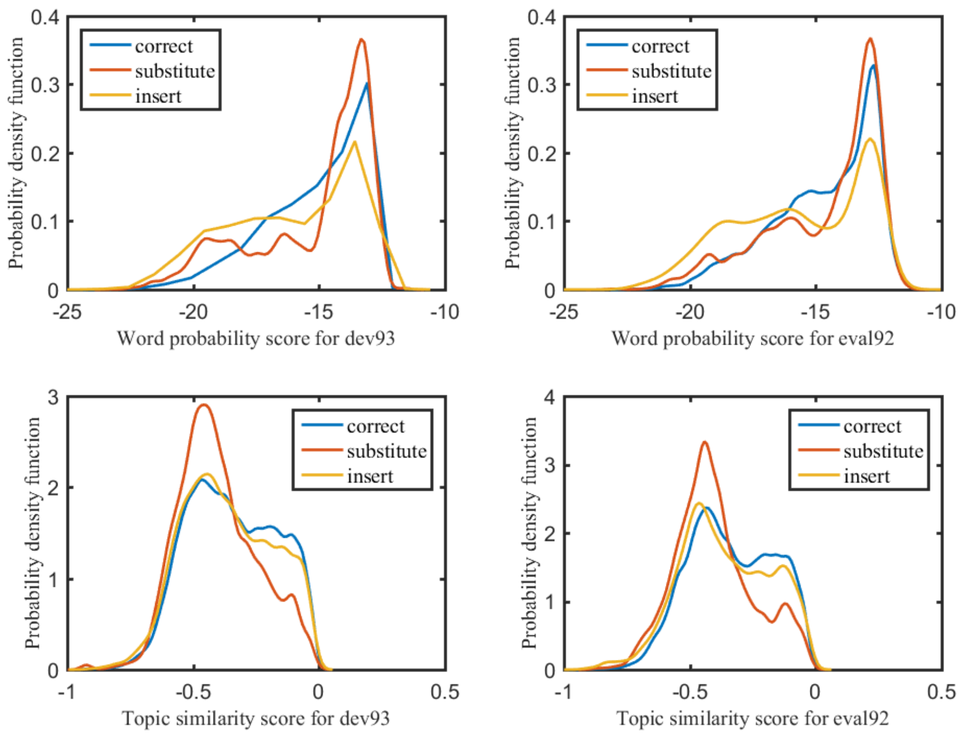


Figure 5. Distribution function of LDA based scores for words with “c”, “s” and “i” tags.

Table 1. The average scores for words with "c", "s" and "i" tags for Dev93 and Eval92 testsets.

Score-Type	Dev93-c	Dev93-s	Dev93-i	Eval92-c	Eval92-s	Eval92-i
word probability	−15.16	−15.20	−16.08	−14.68	−14.68	−15.68
topic similarity	−0.35	−0.40	−0.36	−0.34	−0.40	−0.36

The word error rate (WER) of the new best hypothesis sentences rescored by LDA-based scores are listed in Table 2. "Baseline" here means the WER of the original recognition system, where corresponds to Equation (18). As mentioned before, the word-pair probability score can be regarded as the smoothed 2-gram language model. So we list the result of the 2-gram language model, which is trained by SriLM on the large text corpus. "smallRNN" here refer to a 2-layer RNN with 100 hidden units each layer. In this experiment, α described in Equation (19) is around 0.95 and k is 1 for LDAprob without fallibility weight and 0.03 with fallibility. And α is 25 and 1.5 for LDAtopsim without fallibility and with fallibility respectively. The best result of LDAprob without fallibility weight for Dev93 and Eval92 testset are 9.38% and 6.59% respectively, and are comparable to a small scale RNN. The best result of LDAtopsim for Dev93 and Eval92 testset without fallibility are 9.45% and 6.70% respectively, both of which are higher than the LDAprob score. From the table, we notice that the number of topics, which is a hyperparameter in the topic model, has a strong effect on WER. For LDAprob score, 20 and 30 is a better choice for Dev93, while a topic model with only 10 topics leads to the best result for Eval92. With respect to LDAtopsim, the best choices for the both testsets are 10. We have conducted experiments on topic models of larger number of topics, and find that when n is more than 30, the result is worse as n increases. That is probably because that when n is large, the difference between topics is small. Moreover, for actual speech, if topics are similar, it is easy to transfer between topics. Thus, the larger topic model may reduce the score of correct words together with the incorrect words, and decrease the difference between correct and incorrect words. The fallibility weight reduces WER for all the features. For a topic similarity score with 40-dimension LDA, the fallibility weight reduces WER at 0.16% and 0.21% absolutely for the two testsets respectively from the same feature without fallibility weight. In addition, with fallibility, the rescoring results are less sensitive to the hyperparameter of a number of topics.

Table 2. Word error rate (WER) results of rescoring by LDA sentence probability score.

Score-Type	#Topics	Fallibility	Dev93 WER	Eval92 WER
baseline	-	-	9.58	6.98
2gram	-	-	9.57	6.88
smallRNN	-	-	9.44	6.57
LDAprob	10	no	9.44	6.59
LDAprob	10	yes	9.35	6.52
LDAprob	20	no	9.38	6.65
LDAprob	20	yes	9.35	6.52
LDAprob	30	no	9.39	6.66
LDAprob	30	yes	9.38	6.52
LDAprob	40	no	9.44	6.66
LDAprob	40	yes	9.36	6.54
LDAtopsim	10	no	9.53	6.70
LDAtopsim	10	yes	9.40	6.54
LDAtopsim	20	no	9.45	6.73
LDAtopsim	20	yes	9.38	6.61
LDAtopsim	30	no	9.47	6.75
LDAtopsim	30	yes	9.38	6.57
LDAtopsim	40	no	9.50	6.75
LDAtopsim	40	yes	9.34	6.54

5.3. Word Embedding Based Rescoring

We use word embeddings to re-rank the hypotheses list in this section. Figure 6 is similar to Figure 4. In this figure, we show the average sentence word-pair score and word-discourse score at different ranks. Apparently, the sentences ranked at the front score higher than the sentences ranked behind, especially for the sentences ranked at the first 10 places. This shows that the word-pair score and word-discourse score reflect the basic trends of the sentence.

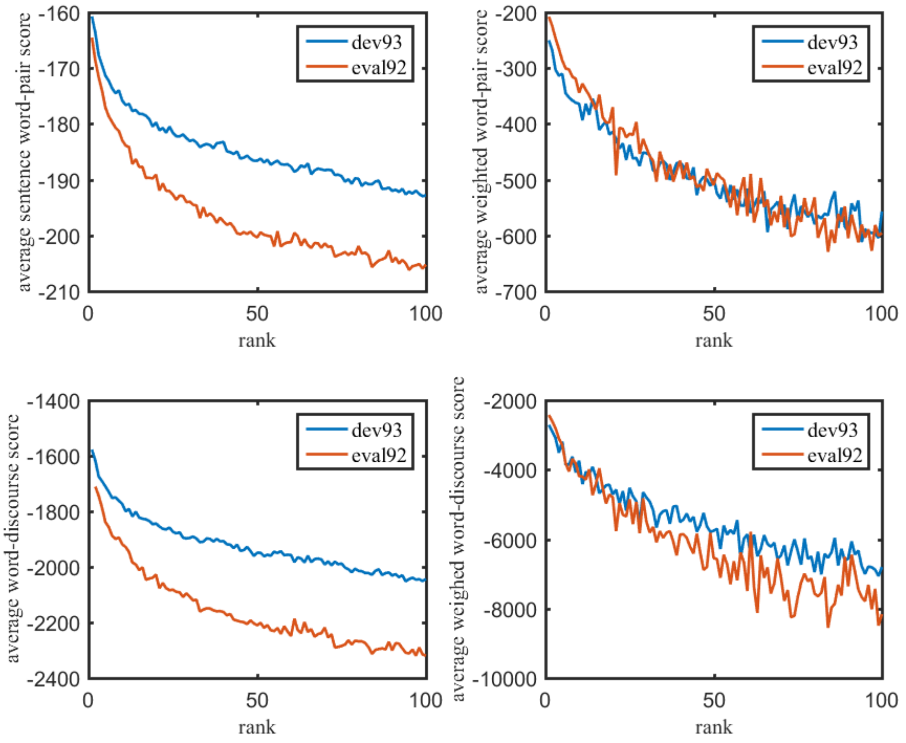


Figure 6. The average score of word embedding based methods for hypotheses. The horizontal axis, raning from 1 to 100, represents the rank of hypotheses. The vertical axis represents the average score of hypotheses at the corresponding rank.

Figure 7 shows the probability distribution of ‘correct’ tags, ‘substitute’ tags and ‘insert’ tags respectively like Figure 5. For both scores and both testsets, words with ‘correct’ tags tends to score higher. The average scores of words with the three tags are listed in Table 3.

The results of the word embedding based word-pair probability score and word-discourse probability score are listed in Table 4. In this experiment, α is around 0.95, which is the same as LDA based scores. k is around 1 for word-pair and word-discourse scores without the fallibility score and it is about 0.05 and 0.01 for a word-pair and word-discourse score with fallibility score respectively. Firstly, we can see from the table that features with all dimensions tested in the experiment reduce WER. As for the LDA based features, features with larger dimension do not result in more significant WER reductions. The best results for Dev93 and Eval92 testsets without fallibility weight are both from the 50-dimension GloVe feature, which are 0.20% and 0.44% absolute WER reduction respectively for word-pair score. For the word-discourse score, the 300-dimension GloVe performs well on the whole of the two testsets, which leads to 0.24% and 0.56% absolute WER reductions on Dev93 and Eval92 testsets respectively compared with the 1-best hypotheses. The fallibility weight reduces WER for most features. Similar to the effect on LDA-based scores, fallibility decreases the performance gap among features dimensions for both word-embedding based scores.

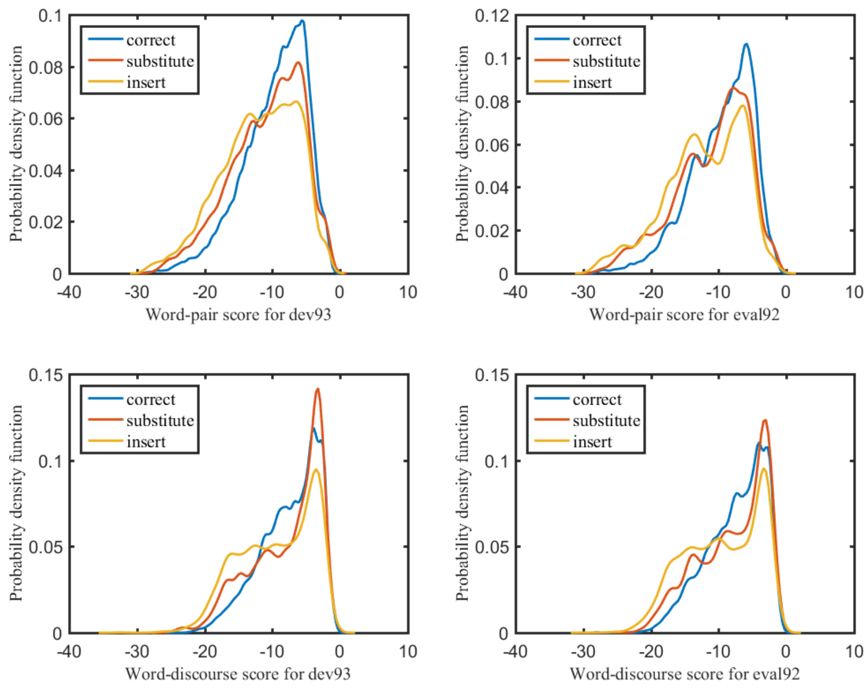


Figure 7. Distribution function of word embedding based scores for words with "c", "s" and "i" tags.

Table 3. The average scores for words with "c", "s" and "i" tags for Dev93 and Eval92 testsets.

Score-Type	Dev93-c	Dev93-s	Dev93-i	Eval92-c	Eval92-s	Eval92-i
word-pair	−9.64	−10.87	−11.98	−9.61	−11.02	−12.15
word-discourse	−7.45	−7.94	−9.39	−7.49	−8.02	−9.47

Table 4. WER results of rescoring by word embedding word-pair probability score.

Score-Type	Feature Dimension	Fallibility	Dev93 WER	Eval92 WER
baseline	-	-	9.58	6.98
2gram	-	-	9.57	6.88
smallRNN	-	-	9.44	6.57
word-pair	30	no	9.44	6.57
word-pair	30	yes	9.33	6.45
word-pair	50	no	9.38	6.54
word-pair	50	yes	9.30	6.50
word-pair	80	no	9.40	6.57
word-pair	80	yes	9.33	6.52
word-pair	100	no	9.44	6.63
word-pair	100	yes	9.34	6.54
word-pair	300	no	9.53	6.91
word-pair	300	yes	9.39	6.68
word-discourse	30	no	9.46	6.61
word-discourse	30	yes	9.30	6.47
word-discourse	50	no	9.34	6.66
word-discourse	50	yes	9.29	6.47
word-discourse	80	no	9.39	6.56
word-discourse	80	yes	9.30	6.52
word-discourse	100	no	9.34	6.70
word-discourse	100	yes	9.32	6.54
word-discourse	300	no	9.34	6.42
word-discourse	300	yes	9.33	6.52

In addition, we combine the LDA-based scores and word embedding based scores to discuss whether the scores can complement each other. We use the following equation to substitute Equation (19) to use the combination of the two kinds of scores:

$$Score(s) = Score_{AM}(s) + \lambda\{\alpha Score_{LM}(s) + (1-\alpha)[\frac{1}{2}\cdot k_1 \cdot Score_{LDA}(s) + \frac{1}{2}\cdot k_2 \cdot Score_{embedding}(s))]\}, \quad (20)$$

where k_1 and k_2 are the parameters selected in last experiments for each scores respectively. Related WER results are listed in Table 5.

Table 5. WER results of the combination of LDA-based scores and word embedding based scores.

LDA Scoretype	Word Embedding Score Type	Fallibility	Dev93 WER	Eval92 WER
baseline	-	-	9.58	6.98
LDAprob10	word-pair30	yes	9.32	6.56
LDAprob10	word-discourse50	yes	9.30	6.54
LDAtopsim40	word-pair30	yes	9.32	6.52
LDAtopsim40	word-discourse50	yes	9.32	6.42

The feature dimensions used here are chosen by consideration of both testsets. HOwever, combining the LDAtopsim score and word-discourse score can reduce the WER of Eval92, compared with using only the 40-topic LDAtopsim score or 50-dimension word-discourse score. On the whole, combining the two kinds of score does not obviously further improve the model, which shows that the proposed LDA-based scores and word embedding based scores contain similar information.

6. Conclusions

In this paper we have presented a new way to use semantic information to rescore the n-best hypotheses without neural network language models. Two kind of semantic features, namely LDA topic features and GloVe continuous word representations, are used and tested in this work. We have designed four sentence coordination scores from the two semantic features. Among the four scores, two are inspired by the LSM and LBL language model respectively, and the other two are first proposed in this paper. In addition, we have designed a fallibility weight to assist the computation of the sentence semantically coordination score by instructing the model to pay more attention to the words that may have more possible choice.

Experiments show that a 10- or 20-dimension LDA based word probability score with fallibility weight reduces WER of Dev93 and Eval92 testsets by absolute values of 0.23% and 0.46% respectively. For word embedding features, the 50-dimension word-discourse probability score with fallibility and 300-dimension word-discourse probability score without fallibility make the most WER reductions for Dev93 and Eval92 testset respectively at 0.29% and 0.51%. Fallibility weight works well on most features for both LDA and word embedding based scores. Although it only reduced WER by a small scale, it strongly decreased the performance gap among features dimensions for both word-embedding and LDA based scores.

As future work, we plan to explore higher-order embeddings which can represent more words rather than only one. On the other hand, the effect of the smoothing parameter γ will be explored in the future work. In addition, the fallibility weight can be used in other rescoring experiments.

Author Contributions: C.L. contributed to the idea of this paper, designed the algorithms and writed the original draft. C.L., T.L. P.Z. was responsible for performing the experiments and the analysis. T.L. and P.Z. reviewed and edited the paper. Y.Y. was in charge of supervision and project administration.

Funding: This work is partially supported by the National Natural Science Foundation of China (Nos. 11590774, 11590770), the Key Science and Technology Project of the Xinjiang Uygur Autonomous Region (No. 2016A03007-1).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mikolov, T.; Karafiát, M.; Burget, L.; Černockỳ, J.; Khudanpur, S. Recurrent neural network based language model. In Proceedings of the Eleventh Annual Conference of the International Speech Communication Association, Chiba, Japan, 26–30 September 2010.
2. Mikolov, T.; Kombrink, S.; Burget, L.; Černockỳ, J.; Khudanpur, S. Extensions of recurrent neural network language model. In Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, 22–27 May 2011; pp. 5528–5531.
3. Tam, Y.C.; Schultz, T. Unsupervised language model adaptation using latent semantic marginals. In Proceedings of the Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, 17–21 September 2006.
4. Mnih, A.; Hinton, G. Three new graphical models for statistical language modelling. In Proceedings of the 24th International Conference on Machine Learning, Corvallis, OR, USA, 20–24 June 2007; pp. 641–648.
5. Arora, S.; Li, Y.; Liang, Y.; Ma, T.; Risteski, A. Linear algebraic structure of word senses, with applications to polysemy. *Trans. Assoc. Comput. Linguist.* **2018**, *6*, 483–495. [CrossRef]
6. Chu, S.M.; Mangu, L. Improving arabic broadcast transcription using automatic topic clustering. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 4449–4452.
7. Jin, W.; He, T.; Qian, Y.; Yu, K. Paragraph vector based topic model for language model adaptation. In Proceedings of the Sixteenth Annual Conference of the International Speech Communication Association, Dresden, Germany, 6–10 September 2015.
8. Lau, J.H.; Baldwin, T.; Cohn, T. Topically driven neural language model. *arXiv* **2017**, arXiv:1704.08012.
9. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
10. Mikolov, T.; Zweig, G. Context dependent recurrent neural network language model. In Proceedings of the 2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, 2–5 December 2012; Volume 12, p. 8.
11. Tam, Y.C.; Schultz, T. Dynamic language model adaptation using variational Bayes inference. In Proceedings of the INTERSPEECH 2005—Eurospeech, 9th European Conference on Speech Communication and Technology, Lisbon, Portugal, 4–8 September 2005.
12. Haidar, M.A.; O'Shaughnessy, D. Novel weighting scheme for unsupervised language model adaptation using latent Dirichlet allocation. In Proceedings of the Eleventh Annual Conference of the International Speech Communication Association, Chiba, Japan, 26–30 September 2010.
13. Haidar, M.A.; O'Shaughnessy, D. LDA-based LM adaptation using latent semantic marginals and minimum discriminant information. In Proceedings of the 20th European Signal Processing Conference (EUSIPCO), Bucharest, Romania, 27–31 August 2012; pp. 2040–2044.
14. Ramabhadran, B.; Siohan, O.; Sethy, A. The IBM 2007 speech transcription system for European parliamentary speeches. In Proceedings of the 2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU), Kyoto, Japan, 9–13 December 2007; pp. 472–477.
15. Heidel, A.; Lee, L.S. Robust topic inference for latent semantic language model adaptation. In Proceedings of the 2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU), Kyoto, Japan, 9–13 December 2007; pp. 177–182.
16. Helmke, H.; Rataj, J.; Mühlhausen, T.; Ohneiser, O.; Ehr, H.; Kleinert, M.; Oualil, Y.; Schulder, M.; Klakow, D. Assistant-based speech recognition for ATM applications. In Proceedings of the 11th USA/Europe Air Traffic Management Research and Development Seminar (ATM2015), Lisbon, Portugal, 23–26 June 2015.
17. Kleinert, M.; Helmke, H.; Ehr, H.; Kern, C.; Klakow, D.; Motlicek, P.; Singh, M.; Siol, G. Building Blocks of Assistant Based Speech Recognition for Air Traffic Management Applications. In Proceedings of the European Union, Eurocontrol-Conference: SESAR Innovation Days 2018, SESARJU, Salzburg, Austria, 3–7 December 2018; number CONF.
18. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 3111–3119.
19. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.

20. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
21. Arora, S.; Li, Y.; Liang, Y.; Ma, T.; Risteski, A. A latent variable model approach to pmi-based word embeddings. *Trans. Assoc. Comput. Linguist.* **2016**, *4*, 385–399. [CrossRef]
22. Hashimoto, T.B.; Alvarez-Melis, D.; Jaakkola, T.S. Word embeddings as metric recovery in semantic spaces. *Trans. Assoc. Comput. Linguist.* **2016**, *4*, 273–286. [CrossRef]
23. Audhkhasi, K.; Sethy, A.; Ramabhadran, B. Semantic word embedding neural network language models for automatic speech recognition. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016; pp. 5995–5999.
24. He, T.; Xiang, X.; Qian, Y.; Yu, K. Recurrent neural network language model with structured word embeddings for speech recognition. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, 19–24 April 2015; pp. 5396–5400.
25. Wagner, R.A.; Fischer, M.J. The string-to-string correction problem. *J. ACM* **1974**, *21*, 168–173. [CrossRef]
26. Povey, D.; Ghoshal, A.; Boulianne, G.; Burget, L.; Glembek, O.; Goel, N.; Hannemann, M.; Motlicek, P.; Qian, Y.; Schwarz, P.; et al. The Kaldi speech recognition toolkit. In Proceedings of the IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, Waikoloa, HI, USA, 11–15 December 2011; number EPFL-CONF-192584.
27. Stolcke, A. SRILM-an extensible language modeling toolkit. In Proceedings of the Seventh International Conference On Spoken Language Processing, Denver, CO, USA, 16–20 September 2002.