- **You may answer in English, Norwegian, Danish or Swedish.**
- **You should answer all questions. The weights of the various questions are indicated.**
- **You should read through the whole set to see whether anything is unclear so that you can ask your questions to the teachers when they arrive.**
- **If you think some assumptions are missing, make your own and explain them!**

1 Machine Translation Evaluation (20%)

To rapidly develop statistical machine translation systems, it is important to have tools for automatic evaluation of the systems. A famous such tool is the BLEU metric which can be expressed by either of the following two equivalent formulas.

$$\text{BLEU-}n = \exp\left(\min(1 - \frac{r}{c}, 0) + \sum_{i=1}^{n} \frac{1}{n} \ln p_i\right)$$

$$\text{BLEU-}n = \min(e^{(1-\frac{r}{c})}, 1) \times (\prod_{i=1}^{n} p_i)^{\frac{1}{n}}$$

(a) Explain the main ideas of the BLEU metric and in particular explain what p_i, n, c and r in the formula stand for.

Solution BLEU is used for evaluating an MT-system, M, which translates from a source language, SL, to a target language, TL. One uses a text, ST, from the source language for which one has one or more manually crafted reference translations in the target language, $R_1, R_2, \ldots, R_n$, for $n \geq 1$. The system produces a candidate translation of ST into TL, call it C, which is then compared to $R_1, R_2, \ldots, R_n$.

BLEU can be considered a modification of word-based precsision and recall. It modifies it in two important respects. First, it does not only consider which words that are common to C and the R_j-s; it also considers longer n-grams. In the formula for BLEU-n, n is a parameter which in principle may be any number, but it is typically selected to be 3, 4, 5, or 6. BLEU-n considers words, bigram, trigram, etc. up to and including n-grams of length n.

Second, BLEU is based only on precision, it does not consider recall. It only considers whether n-grams in C can be found in corresponding sentences in one of the R_j-s, but not the other way around.

p_i is the n-gram precision for n-grams of length i. It can be expressed by

$$\frac{\sum_{s \in C}(\sum_{n\text{-}gram \in s} Count_{clip}(i\text{-}gram, s, s_j))}{\sum_{s \in C}(\sum_{n\text{-}gram \in s} Count(i\text{-}gram, s))}$$

The denominator counts the total number of i-grams in C, while the numerator counts how many of these are found in at least one of the R_j-s. Both counts are done for the whole corpus, i.e., it is calculated for each sentence s and then added over all the sentences. Special care must be taken for i-grams that occur more than one time in C, say k times. The number of occurrences is then counted m times where m is the smaller number of k and the number of occurrences of this i-gram in the s_j where it occurs the most times.

Since BLEU does not consider recall, it uses a penalty for short translations, calculated by r and c. Here c is the length of C, i.e., the sum of the lengths of the sentences in C. To calculate r, for each sentence, s_m, in C, choose the corresponding sentence r_m from the R_j-s that is most similar to s_m in length. Then r is the sum of the lengths of all the r_m-s.

(b) Assume an MT system produces the candidate translation

(1) the elk was chasing the little rat

and it should be BLEU evaluated against the two reference translations

(2) a big moose was chasing the little mouse

(3) the big elk was following a strange little brown animal

What are the values of p_1, p_2, p_3, p_4, c, r in this example?
(You do not have to calculate the full BLEU score).

- $c =$ the length of sentence (1) $= 7$
- $r =$ the length of sentence (2) $= 8$
- $p_4 = \frac{1}{4}$ There is only one 4-gram: *was chasing the little*
- $p_3 = \frac{2}{5}$ *was chasing the* and *chasing the little*
- $p_2 = \frac{4}{6}$ *elk was* and *was chasing* and *chasing the* and *the little*
- $p_1 = \frac{5}{7}$ *the* counts once

2 Phrase-based Translation (30%)

We are going to translate the following Norwegian sentence into English using phrase-based statistical machine translation.

(4) elgen jagde musa

We assume the following simple phrase translation table (all other alternatives have probability 0):

$\phi(\text{elgen} \mid \text{the moose}) = 0.5$
$\phi(\text{elgen} \mid \text{the elk}) = 0.3$
$\phi(\text{jagde} \mid \text{chased}) = 0.3$
$\phi(\text{jagde} \mid \text{was chasing}) = 0.3$
$\phi(\text{musa} \mid \text{the mouse}) = 0.5$

To simplify, we will in the following only consider the translation alternatives without reordering.

(a) We will use beam search for the decoding. Draw the search graph for the decoding algorithm presupposing no reordering (and no recombination).

See drawing!

(b) One technique for reducing the search space is called hypothesis recombination. Redraw the search graph with recombination using a bigram language model. What are the requirements for recombination?

See drawing!

The requirement for recombination is that the two hypotheses cover the same part of the input sentence, and that the $n-1$ last words of the output hyptheses are the same assuming an n-gram language model. Thus, with a bigram model this requirement amounts to that the last words in the two output hypotheses are identical.

(c) Assume the following from the language model where $p(w_2 \mid w_1)$ is the probability of w_2 being the next word after w_1, and # indicates start or end of sentence.

$p(\text{the} \mid \#) = 0.1$
$p(\text{elk} \mid \text{the}) = 0.0002$
$p(\text{moose} \mid \text{the}) = 0.0001$
$p(\text{was} \mid \text{elk}) = 0.1$
$p(\text{was} \mid \text{moose}) = 0.1$
$p(\text{chased} \mid \text{elk}) = 0.01$
$p(\text{chased} \mid \text{moose}) = 0.01$
$p(\text{chasing} \mid \text{was}) = 0.001$
$p(\text{the} \mid \text{chased}) = 0.5$
$p(\text{the} \mid \text{chasing}) = 0.5$
$p(\text{mouse} \mid \text{the}) = 0.001$
$p(\# \mid \text{mouse}) = 0.1$

What will be the highest ranked candidate translation? State reasons for your answer.

There are 4 candidate translations:

1. the moose chased the mouse
2. the moose was chasing the mouse
3. the elk chased the mouse
4. the elk was chasing the mouse

We first compare (1) to (3). All probabilities are the same except for two; the translation probabilities for *the elk* vs. *the moose* and bigram probabilities for $p(\text{elk} \mid \text{the})$ vs. $p(\text{moose} \mid \text{the})$.

We see that $\phi(\text{elgen} \mid \text{the moose}) \times p(\text{moose} \mid \text{the}) = 0.5 \times 0.0001 = 0.00005$ while $\phi(\text{elgen} \mid \text{the elk}) \times p(\text{elk} \mid \text{the}) = 0.3 \times 0.0002 = 0.00006$. Hence (3) is preferred to (1). By the same reasoning (4) is preferred to (2).

Comparing (3) to (4), we again consider the parts of the calculation that are different. $\phi(\text{jagde} \mid \text{chased}) \times p(\text{chased} \mid \text{elk}) \times p(\text{the} \mid \text{chased}) = 0.3 \times 0.01 \times 0.5 = 0.0015$ while $\phi(\text{jagde} \mid \text{was chasing}) \times p(\text{was} \mid \text{elk}) \times p(\text{chasing} \mid \text{was}) \times p(\text{the} \mid \text{chasing}) = 0.3 \times 0.1 \times 0.001 \times 0.5 = 0.000015$. Hence, (3) is the preferred reading.

3 Semantic models in general (10%)

How do you understand the difference between *distributional* and *distributed* word meaning models? Can a model be both *distributional* and *distributed*?

A model can be both distributional and distributed: in fact, any word embedding model with reduced dimensionality is an example of this. All corpus-based word meaning models are 'distributional' in the sense that they are derived from *distributions* of word co-occurrences in the underlying corpora. 'Distributed' means something else: if we use dense word vectors (hundreds of components), there is little chance for the components of these vectors to be directly mapped to some semantic 'features'. Instead, words and concepts are represented with *combinations* of the components' values, or *multiple neuron activations*, using the artificial neural networks terminology. Each semantic feature (for example, 'the property of being an animal') is *distributed* among many components, and each component (neuron) participates in representing many different features.

This is not the case for the standard count-based models without dimensionality reduction: in them, each vector component is directly mapped to a particular context word. However, prediction-based models, like CBOW or Continuous SkipGram are both distributional and distributed.

4 Count-based models (20%)

You are given a symmetric matrix **M** with the normalized values of word co-occurrences frequency in some text corpus:

	computer	*laptop*	*mainframe*	*small*	*puppy*	*dog*
computer	2	2	1	0	0	0
laptop	2	2	0	1	0	0
mainframe	1	0	2	0	0	0
small	0	1	0	2	1	0
puppy	0	0	0	1	2	2
dog	0	0	0	0	2	2

4.1 Calculating similarities

Each row in **M** is a distributional vector for the corresponding word. Rank vectors for all the words by their cosine distances to the vector of the word '*puppy*'. Report the distances as well (you can provide approximate values, no need for extreme precision).

'puppy' vector: [0, 0, 0, 1, 2, 2] (unit-normalized: [0, 0, 0, $\frac{1}{3}$, $\frac{2}{3}$, $\frac{2}{3}$])

Similarities to 'puppy':

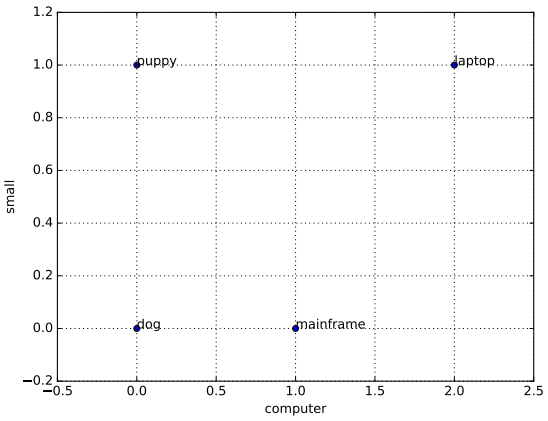
1. **dog** ($\frac{8}{3\sqrt{8}} \approx 0.94$)

2. **small** ($\frac{4}{3\sqrt{6}} \approx 0.54$)
3. **laptop** ($\frac{1}{9} \approx 0.11$)
4. **computer** (0)
5. **mainframe** (0)

4.2 Visualizing vectors

It is often useful to 'embed' semantic vectors of words into a **2-dimensional space** to visualize how they are related to each other. Use the words '*computer*' and '*small*' as the horizontal and vertical dimensions and embed the vectors of all the other words into this 2D space. Plot it.

One should simply use the co-occurence counts with these two words as the horizontal and vertical axis. The resulting plot should look approximately like this:



5 Prediction-based models (10%)

Suppose you are training a **CBOW** model with the vector size 300 on a corpus containing 100 000 word types and 100 000 000 word tokens. You chose 5 words as the width of your symmetric window.

1. What will be the dimensionalities (shapes) of the *input matrix* and the *output matrix* in your neural network architecture?

   The input matrix shape will be 100000:300, and the output matrix shape will be 300:100000.

2. At training time, what is the *input* to the neural network at each prediction step? Give an example of a single prediction with any real sentence.

   In the CBOW algorithm, the input at the prediction step is the sum or the average of the vectors (embeddings) of the left and right neighbors of the current word. Let's consider the sentence *'The cat sits on a mat'*. Suppose our symmetric window size is 1 (one word to the left and one word to the right), and we are now at the word *'sits'*. Then what we feed to the predictor is the current **input vectors** for the words *'cat'* and *'on'* (averaged or summed). We then calculate the cosine similarities of the resulting 'context vector' to the current **output vectors** of all the words in the vocabulary. These similarities are transformed into the probability distribution using softmax.

   After that, we can easily calculate the prediction error on each word (the 'correct' prediction is 1 for *'sits'* and 0 for all other words). This error is then back-propagated to the output and the input matrices, thus shifting input vectors for *'cat'* and *'on'* a bit closer to the output vector of *'sits'*. Then we move on to the next word.

6 Comparing distributional models (10%)

You are studying *semantic shifts* that words undergo over time. You trained several distributional models on corpora of texts produced in different decades of the XX century. Enumerate possible ways to discover the *degree of semantic change* for some word a. Suppose that a is present in all the models.

Briefly describe pros and cons of each method.

1. Compare ranked lists of a nearest associates in different models (using Jaccard distance or Kendall's τ, etc). This is easy and does not demand any complex calculations. However, this comparison is indirect and you can't extract really complex quantitative laws of diachronic change this way.

2. Compile a short list of words for which you are sure that their meaning remains stable across time. Use this list to learn a linear transformation matrix from all models to some 'proxy' model (chosen arbitrarily from what you have). Then one can 'project' vectors of a from different models to this 'proxy' semantic space, thus making them directly comparable (by cosine similarity). This allows to perform full-scale vector space analysis, but the list of 'stable' words is needed, which can be difficult to compile.

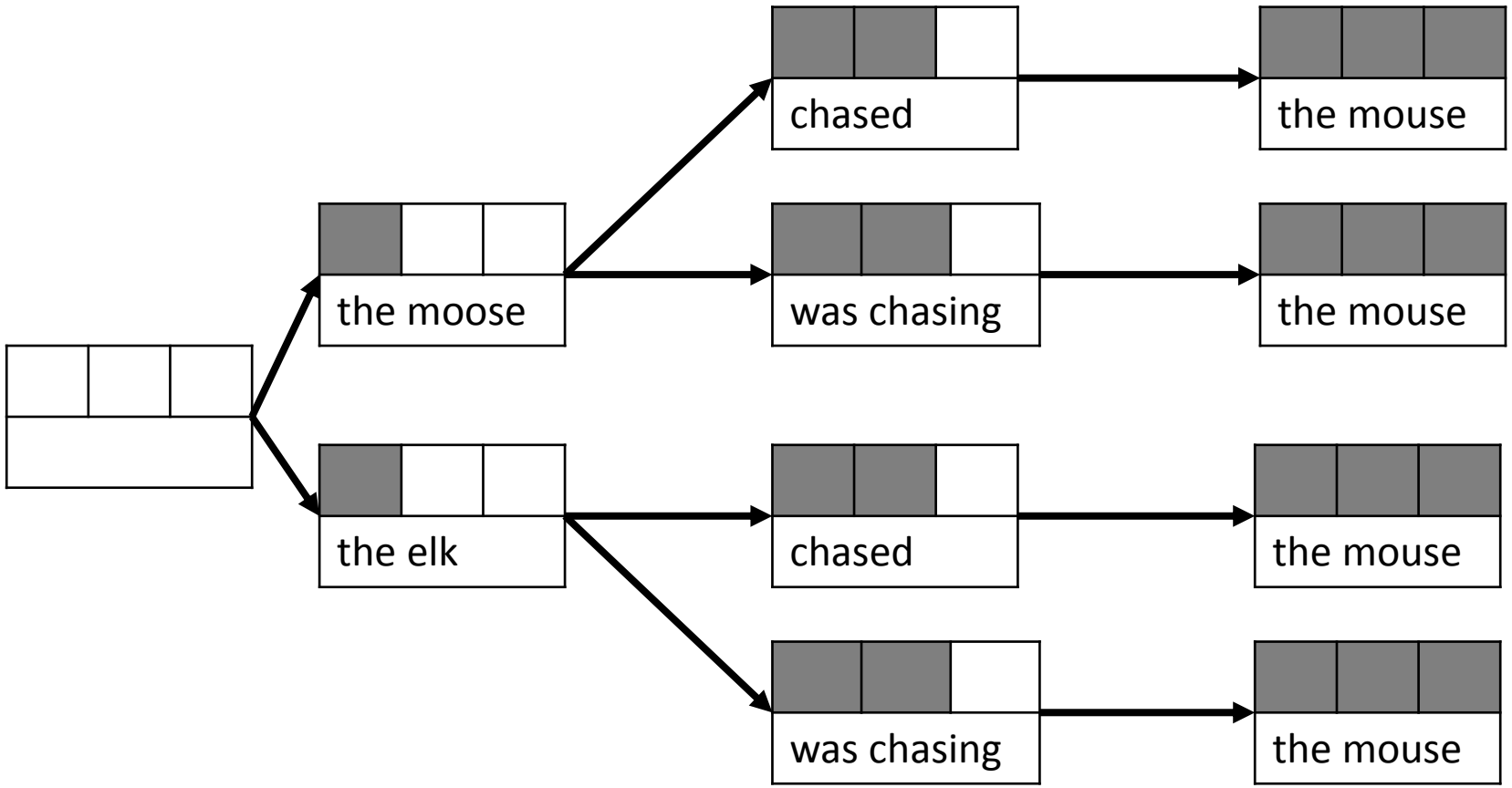
3. 'Squeeze' the whole semantic space of all models into the 'proxy' semantic space, for example using the *orthogonal Procrustes transformation*. Then they essentially become one model, and one can easily measure whatever distances one wants between any pairs of words (for example a_{60s} and a_{90s}). With this approach, it is possible to perform full-scale analysis without any *ad hoc* word lists: the algorithm will just try to rotate and scale the spaces to make them maximally like each other without losing the pairwise distances between words. However, this can result in omitting some important non-trivial differences between the semantic spaces. Also, the transformation itself is rather computationally expensive.

4. etc...

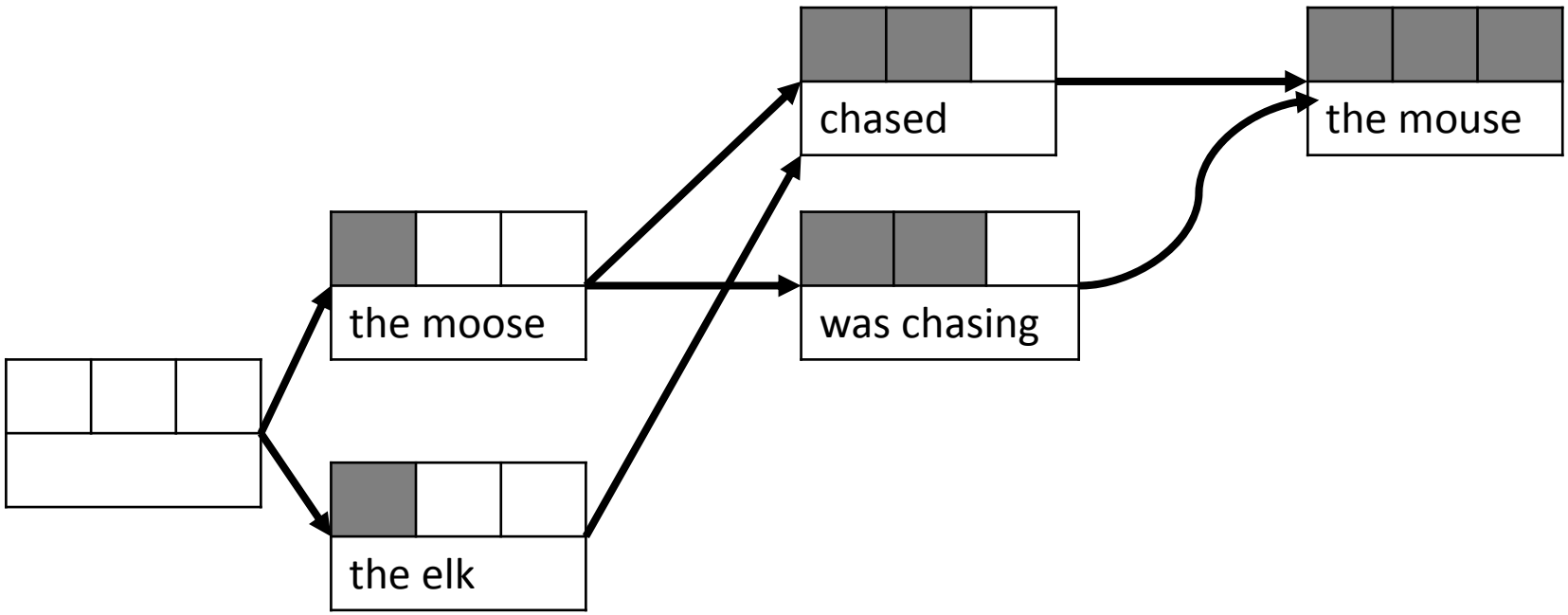
(These variants are basic, but the students can easily come up with something else as well.)

END

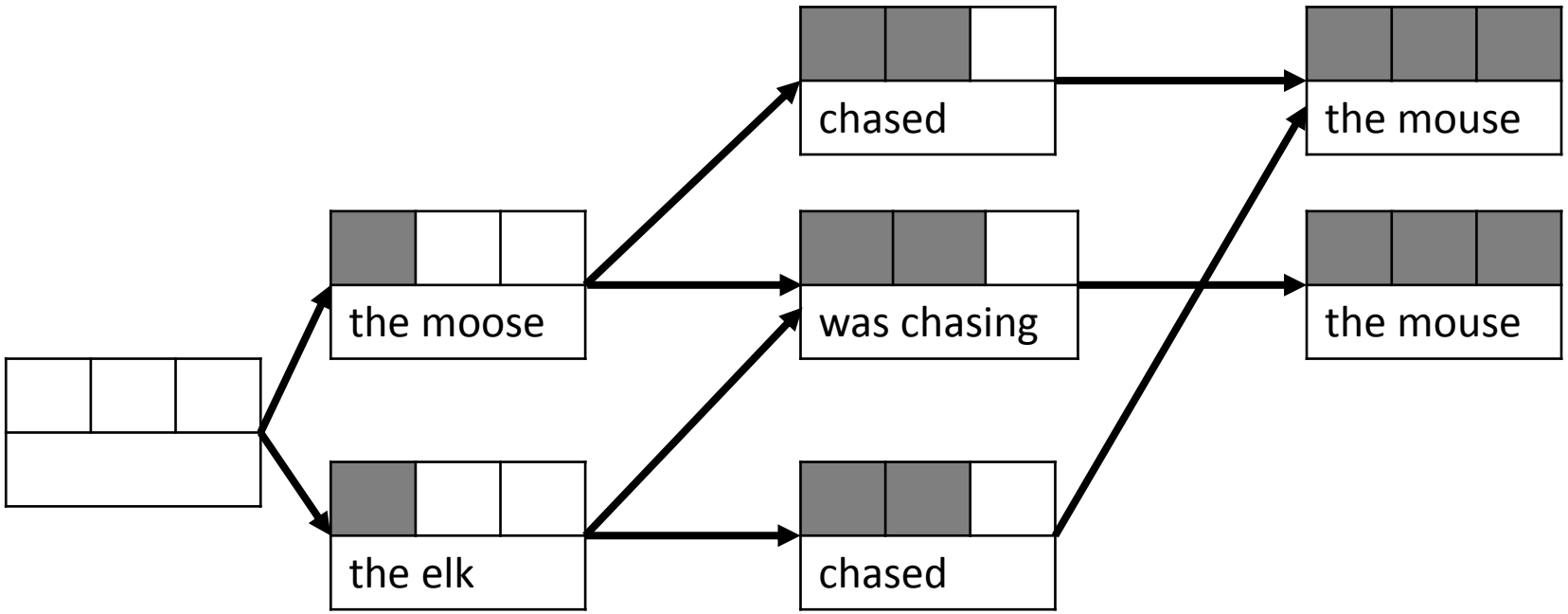
Without recombination and reordering



Recombination, bigram language model



Recombination: trigram language model



First step, general case (with reordering)