



UNIVERSITÄT KARLSRUHE (TH)
INSTITUT FÜR THEORETISCHE INFORMATIK
BUNDESREPUBLIK DEUTSCHLAND

Carnegie Mellon University
Language Technologies Institute
Pittsburgh, PA, U.S.A.



*International center for
Advanced Communication Technologies*
<http://interact.ira.uka.de>

LATTICE EVALUATION TECHNIQUES

APPLIED TO DOMAINS OF

STATISTICAL MACHINE TRANSLATION

Diploma Thesis
by
Dietmar Bernreuther

September 15, 2005

Supervisors:
Dipl.-Phys. M. Phil. Stephan Vogel (☞)
Dipl.-Biochem. Muntzin Kolss (☐)
Prof. Dr. rer. nat. Alex Waibel (☐, ☞)

To my parents

Declarations

I hereby confirm that I wrote this thesis by myself and without undue foreign help. The used literature is completely listed in the bibliography.


Karlsruhe, September 15th, 2005



Dietmar Bernreuther

Permission is herewith granted to the Universität Karlsruhe (TH), Germany, and the Carnegie Mellon University in Pittsburgh, PA, USA, to circulate copies of this thesis for non-commercial purposes.

Karlsruhe, September 15th, 2005



Dietmar Bernreuther

Acknowledgements

Thanks in particular go to my supervisor Stephan Vogel for his patience and constructive support, also for showing me that there is no reason for losing one's good mood when s. . . happens; to our team at CMU and UKA for help, support and collegueship; to KJ, Flee, Sara(h), Aubrey, Steve, Phil, Crazy "Maimun" Alkas, Chris, Alex, "party-hopping" Tim, Nancy, Claudia, Rebekka and all others who gave me a great time in Pgh; family Bazán in particular for a wonderful X-mas; to Tes, kisses.

Special thanks go to my parents for their everlasting support and for having me lead where I am today.

Part of this thesis was written at Carnegie Mellon University, PA, U.S.A. The exchange has been financially supported by interACT exchange.



<http://interact.ira.uka.de>

Universität Karlsruhe

InterACT
School of Computer Science
Margit Rödder
Press and Communication
Am Fasanengarten 5
D-76131 Karlsruhe
Tel: +49 721 608 6385
E-Mail: roedder@ira.uka.de

Carnegie Mellon University

InterACT
School of Computer Science
Thomas Schaaf
5000 Forbes Avenue
Pittsburgh, PA, 15213, USA
Tel: +1 412 268 1461
E-Mail: tschaaf@cs.cmu.edu

Last but not least, I would like to thank Prof. Alex Waibel at this point, for supporting and encouraging me to write my thesis abroad, and for helping so many other students to do the same.

Abstract

In a world getting smaller, switching between different languages is an increasingly important skill for students, politicians, commercial agents, internet users or tourists, to just mention a few. While traveling to another country has become an experience affordable by major parts of society, one doesn't have to travel far in order to meet a foreigner who uses a script most people probably wouldn't be able to decipher, less than ever read, be it in the Thai restaurant round the corner or in the Pakistani grocery store down the street. Although knowledge of a foreign language is not a privilege of intellectuals and/or professional interpreters any more, speaking more than one foreign language (if any) still seems to be infeasible for broad parts of society. Furthermore, many — if not most — people of this world do not have access to public education and are therefore handicapped in their ability to face the tasks and benefit from the opportunities of a globalizing planet.

Machine Translation (MT), the field of computer science dealing with the construction of artificial systems and devices for the translation of human language, is a challenging task for computer scientists. The non-formal nature of natural language makes it difficult to extract the information needed to carry the meaning of an expression from one language into another. Various approaches to tackle the problem have been tried in recent decades. Early systems tried to encode the knowledge of an expert linguist (therefore called *knowledge-based* or *expert systems*), into an artificial system. However, such systems turned out to be expensive and inflexible: Few components of a system constructed for one language pair could be re-used for another language pair, or even just the opposite direction.

The rise of machine performance has made an alternative approach possible: In *Statistical Machine Translation (SMT)*, we try to learn the rules that are necessary to translate from one language into another by scanning a large parallel text corpus of the language pair in question. Such parallel corpora already exist (e. g. parliament minutes of transnational organizations, multilingual newspapers, etc.) and it is much easier to acquire such a corpus than to teach a computer the knowledge of a human lingual expert.

In this thesis, we focus on the part of an SMT system that is responsible for the task of actually finding the correct translation, which is called *decoder*. Most contemporary decoders use *translation lattices* in order to represent the search space of possible translations of a sentence. Although lattices are data structures possessing many interesting features worth examining, assessment of translation lattices usually happens indirectly: We can meter the quality of a

translation hypothesis returned by the path search by comparing it with one or more reference translations, using well-known similarity metrics [PRWZ02, NIS]. The resulting score, in turn, can serve as a figure of translation lattice quality. This way, the impact of changing the parameters of a decoder’s lattice generation module is observed at the backend of the decoder, which means that translation lattices can be considered to be some kind of “hidden variable” of the decoder.

In order to reveal the entire information contained in a translation lattice, we will apply various techniques that decouple the generated lattice from the path search. Firstly, we will substitute the model-based path search by a search that does not depend on the statistical models and tuning parameters used by the decoder’s path search. Secondly, we will use statistics computed on translation lattices compared to statistics computed on the reference translations in order to derive a figure of similarity that can be used as a translation lattice evaluation metric.

In the first chapter, we will provide an overview of the scientific field of machine translation. In the second chapter, we will introduce an *oracle decoder* (which searches a path through a lattice with respect to a reference) that has been implemented in the course of this thesis; the method of assessing lattices using oracle experiments has been applied before by other groups [ZN05]. In the third chapter, we will discuss different aspects of *translation lattice quality* in order to outline design criteria for a hypothetical translation lattice evaluation metric. As a result, we will introduce a model-free lattice evaluation metric called *Standard Word Count Distance (SWCD)*, which is, to our knowledge, a completely new approach for the task of dismantling the decoder into its components. As an application, we implemented and tested an oracle pruning method, which is the basis for the training of a lattice pruning module which will eventually be part of our decoder.

Contents

1	Introduction	1
1.1	Machine Translation: System Classification	1
1.2	Statistical Machine Translation	3
1.3	Breaking down the models	6
1.3.1	The noisy channel	6
1.3.2	Translation model	7
1.3.3	Language model	8
1.3.4	Other models	9
1.3.5	Smoothing	10
1.4	Decoding	10
1.5	Evaluation	14
1.6	System overview	14
2	Oracle Decoding	17
2.1	Implementation	19
2.1.1	The Levenshtein decoder	20
2.1.2	The PMED forced aligner	27
2.2	Decoder Analysis: Taking the Decoder Apart	29
2.3	Performance-Density Graphs	32
2.4	Experiments	33
2.4.1	Optimality with respect to the objective function	33
2.4.2	Optimality with respect to the evaluation metric	33
2.4.3	Comparing systems using oracle scores	37
2.5	Conclusions	39
3	Advanced Lattice Evaluation	41
3.1	Translation Lattice Quality	41
3.2	Lattice Evaluation Metrics	43
3.2.1	Standard measures	44
3.2.2	Search algorithm independence	45
3.2.3	Model consistency	47
3.2.4	Graph Error Rate and Graph Word Error Rate	50
3.2.5	The canonical metric	50
3.2.6	Novel approaches	51
3.3	Standard Word Count Distance (SWCD)	54
3.3.1	Multiple references	57
3.3.2	Bounding the redundancy	58
3.3.3	Predicting decoder performance	62

3.3.4	Precision	63
3.3.5	Recall and uniformity	63
3.3.6	Lattice complexity	63
3.3.7	Unigrams and position independence	64
3.3.8	Comparing systems using SWCD	65
3.4	SWCD Oracle Pruning	66
3.4.1	How a single edge changes the metric	66
3.4.2	The pruning criterion	69
3.4.3	Decoder performance using SWCD pruning	71
3.4.4	Introducing a threshold	74
3.4.5	Decoder performance using SWCD threshold pruning	77
3.4.6	SWCD pruning compared to optimal pruning	80
3.5	Conclusions	84
4	Conclusions	87
4.1	Future Objectives	87
4.1.1	Model dependence	87
4.1.2	Translation lattice optimization	88

List of Tables

1.1	Nomenclature of used variable and function names	2
1.2	Training corpora	15
1.3	Test sets	16
2.1	BLEU score by n -gram for various systems	36

List of Figures

1.1	The translation triangle	3
1.2	Number of translation directions using an interlingua	3
1.3	Model of the noisy channel	7
1.4	Word alignment	8
1.5	Smoothing example	11
1.6	Lattice-based decoder	12
1.7	Nonmonotonicity example	13
2.1	Impact of global reordering on Levenshtein distance	19
2.2	Dynamic programming matrix	20
2.3	Dynamic programming: Cell expansion	21
2.4	Dynamic programming: Target word expansion	22
2.5	Levenshtein decoder hypothesis class (UML)	23
2.6	Levenshtein decoder: Hypothesis expansion	24
2.7	Levenshtein decoder: Recombination result	25
2.8	Equivalent decoder lattice representing search space	26
2.9	PMED decoder hypothesis class (UML)	29
2.10	Oracle vs. decoder score	31
2.11	Two decoder PDGs vs. corresponding oracle PDGs	32
2.12	Optimality with respect to the objective function: Levenshtein decoder	34
2.13	Optimality with respect to the objective function: PMED forced aligner	34
2.14	Optimality with respect to the evaluation metric: Levenshtein decoder	35
2.15	Optimality with respect to the evaluation metric: PMED forced aligner	36
2.16	Decoder performance on different trainsets	37
2.17	Levenshtein decoder performance on different trainsets	38
2.18	PMED forced aligner performance on different trainsets	38
2.19	Trouble spots: Comparing tuned vs. untuned	39
3.1	Model consistency	49
3.2	Edge Precision	52
3.3	Varying hit rate interpolation factors for Edge Precision	53
3.4	Components of the Edge Precision metric	53
3.5	A trivial lattice for a single reference sentence.	57
3.6	A trivial lattice for two reference sentences.	57

3.7	An optimal lattice for two reference sentences.	57
3.8	Varying $\underline{\lambda}$ on the English-Spanish test set	59
3.9	Varying $\underline{\lambda}$ on the Spanish-English test set	59
3.10	Varying $\underline{\lambda}$ on the English-French test set	60
3.11	Varying $\underline{\lambda}$ on the French-English test set	61
3.12	Comparing end-to-end decoder performance vs. SWCD	62
3.13	SWCD on different trainsets	65
3.14	SWCD pruning on decoder performance	71
3.15	Decoder with SWCD pruning compared to oracle score	72
3.16	SWCD pruning on oracle score	73
3.17	Relative amount of pruned edges	74
3.18	$\widehat{\Delta\tau}$ with respect to number of nodes and lattice density	75
3.19	$\widehat{\Delta\tau}$ with respect to number of edges	76
3.20	Relative amount of pruned edges using threshold normalization	77
3.21	SWCD pruning with threshold	78
3.22	Using an extremely high threshold	78
3.23	Growth of $\widehat{\tau}'$	79
3.24	SWCD pruning with unnormalized threshold	80
3.25	Decoder with SWCD threshold pruning compared with oracle score	81
3.26	Impact of SWCD threshold on oracle score	82
3.27	Oracle score limiting decoder performance on pruned lattice	83
3.28	Decoder vs. oracle performance for increasing ϑ	83
3.29	Oracle vs. decoder performance for large ϑ	84

Chapter 1

Introduction

Throughout this section, we will assume a Spanish–English text translation task, using the set of symbols listed in table 1.1. We reduce the translation task to the translation of a single sentence.¹ “Sentence” may or may not match the linguistic term here, for instance, we might want to translate only partial sentences or utterances, particularly in Spoken Language Translation (SLT) tasks, where grammatical structures like “sentences” are hard to determine. A more general (and technical) term in this context would be “segment”². Any sentence (or segment) is a somehow “valid” sequence of *words*, which are the smallest entities (“atoms”) subject to translation³.

1.1 Machine Translation: System Classification

Considering the natural process of a translation (that is, through a human translator), we can observe two essential phases during such a translation:

- Interpretation (Analysis) and
- Generation

Firstly, the human translator (or “interpreter”) tries to completely *understand* the source sentence. Formally, we will model this by a family \mathcal{C} of language-independent *concepts*. The translator now maps each sentence onto an abstract

Concepts as a formal way to describe the pragmatics of an expression

¹This is truly a simplification of the translation task: When processing a document, we can usually benefit from the information given in previous (and sometimes also succeeding) sentences for the translation of a single sentence. In some cases, this context information might even be necessary for the correct interpretation of a sentence (for instance for prepositions, which cannot be resolved without knowing the referred words in the preceding sentence(s)), and correct interpretation might be required in order to unambiguously determine genus and numerus in the target language.

²Note that the *segmentation problem*, the problem of splitting a document into linguistic units, might pose a non-trivial task, especially for parallel corpora ([GC91, ZV02, ZZVW03]) or SLT (if not already done by the Automatic Speech Recognition (ASR) engine).

³Other approaches are possible, although rather exotic. Usually it is more convenient to flex the notion “word” and split off interesting particles of a word if necessary, e.g. “unambiguous” \Rightarrow “un + ambiguous”

S	The source language (Spanish), a set of sentences
\mathcal{V}_S	The vocabulary of the source language, a finite set of words
\mathcal{E}	The target language (English)
\mathcal{V}_E	The target vocabulary
\mathcal{L}	A language in general
\mathcal{C}	The family of concepts (idea, intention, etc.)
s	Elements of the source language (a Spanish sentence)
e	An English target sentence respectively
\hat{e}	The best translation for a given source sentence
Ω	A random space
X, Y, Z, \dots	Random variables
Pr	A probability measure
Pr^X	The probability measure implicated by the random variable X
$\text{Pr}(a b)$	The probability of a given b
$\text{Pr}(a, b)$	The joint probability of a and b

Table 1.1: Nomenclature of used variable and function names

concept, which can be — assuming a *deterministic* process in the first place — described by an *interpretation function*.

$$\mathcal{I} : S \longrightarrow C \quad (1.1)$$

After having understood the meaning of the source sentence, the translator consequently will try to express the inherent concept in the target language, (s)he will *generate* a translation:

$$\mathcal{G} : C \longrightarrow E \quad (1.2)$$

Again, we assume a *deterministic* process.

Transfer as additional intermediate step between interpretation and generation

A common error in human translations is not to entirely abstract from the source language. This usually leads to bumpy “word-to-word”-translations, where the vocabulary has been shifted to the target language, the grammatical structure of the source language, however, is still evident. In this case, an additional step called *transfer* is necessary in order to shift the language specific parts of the concept (caused by the incomplete abstraction from the source language) to the target language.

The whole process can be depicted quite ostensively by a *translation triangle* (fig. 1.1). In MT literature, the interpretation step is usually referred to by the more technical term *analysis*. The triangle visualizes the three main classes of MT systems: *direct translation*, *transfer-based translation* and *interlingua-based translation*. Note that instead of the merely theoretical family of concepts, an explicit *Interlingua*, a formal artificial concept description language is placed on top of the triangle. An explicit description language is prerequisite to expressing and eventually processing intermediate results (intermediate translations). However, by using an explicit language, one loses much of the universality of concepts: It is impossible to design a language that can express any abstract concept⁴, and although we can restrict the expressibility of an intermediate lan-

⁴Note that a language over a finite alphabet is always countable, an universal family of concepts is not.

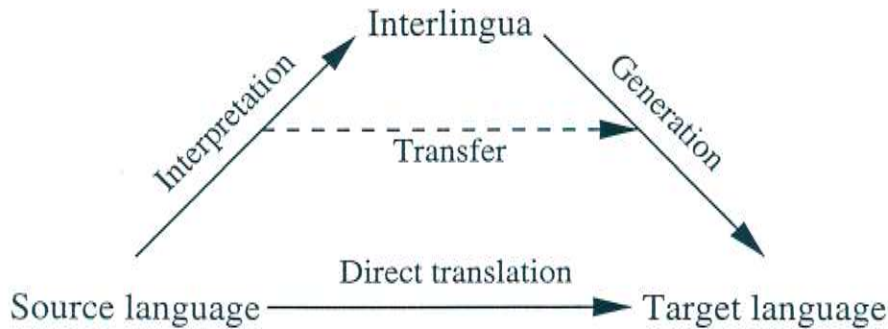


Figure 1.1: The translation triangle [Och02]: Interpretation (or Analysis) — Transfer — Generation

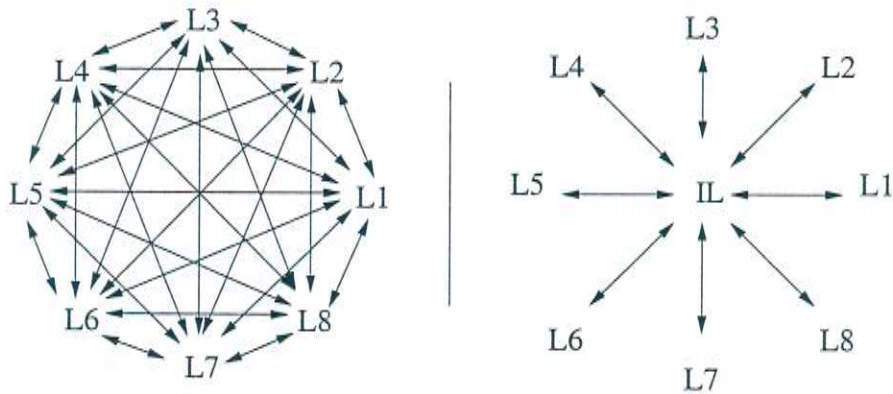


Figure 1.2: The number of possible translation directions drops from $O(n^2)$ to $O(n)$ for a task including n languages.

guage to the expressibility of any language being part of the translation task⁵, formal interlingua are always task- and domain-specific. There is no such thing as a formal universal language that can describe the whole world.

Formal languages are never universal

Apart from the theoretical motivation, there is also a very straightforward reason why to use an interlingua (formal or non-formal): In order to translate between n languages, one only needs $O(n)$ translation directions instead of $O(n^2)$. For any language, one only has to design the interpretation and generation functions in order to being able to translate from and into any language participating in the Interlingua framework.

An interlingua reduces the number of translation directions significantly

1.2 Statistical Machine Translation

The basic approach of *statistical* machine translation is to select the *most probable* English sentence \hat{e} to be the translation of a given source sentence s :

⁵We could, for instance, choose (simplified and/or enriched) English to be a non-formal Interlingua [RW04], assuming that anything we can express in another natural language can be expressed in English as well.

$$\hat{e} = \arg \max_{e \in \mathcal{E}} \Pr(e|s) \quad (1.3)$$

Machine translation as
MAP-classification

This *Fundamental Theorem of Statistical Machine Translation* postulates that we should “Choose an English sentence e as translation of s if, and only if, e maximizes the a-posteriori-probability of e given f ”. This general approach is widely known as Maximum-a-posteriori classification (MAP). $\Pr(e|s)$ is the statistical translation model, the $\arg \max$ operator denotes a search problem. In the context of statistics, we also consider source and target sentences to be statistical *events*.

Consider the following two sentences:

$$\begin{aligned} e_1 &= \text{Where is the shoe shop?} \\ s &= \text{¿Dónde está la zapatería?} \end{aligned}$$

e_1 seems to be a perfect translation of s , so we would expect a translation probability close to one: $\Pr(e_1|s) \approx 1$. However, probabilities have to sum up to one by definition, hence there wouldn't be any probability mass left for other possible translations, for instance

$$e_2 = \text{“Where is the shoe shop, please”}$$

One could, of course, argue that e_2 is not a valid translation of s , as s does not contain a correspondence to the English word “please” (“por favor”). This argumentation, however, does not hold anymore for the perfect translation

$$e_3 = \text{Where is the shoe store.}$$

which should at least have a similar translation probability as e_1 .

Translation probabilities
can become very small

Evidently, the notion “probability” cannot just be interpreted as a general figure of merit for translation quality in this context, assigning values close to 1 to “good” translations and values close to 0 to the “bad” ones. We can rather assume (considering the numerous valid possibilities to translate a sentence) that translation probabilities for valid translations can become arbitrarily small. We can live with that, as long the “bad” translation get even worse scores:

$$1 > \Pr(e_1|s) \gg \Pr(\text{“Your shoe is open.”}|s)$$

Meaning of probability:
relative frequency in a
stochastic process

In order to use probabilistic methods in machine translation, we have to have a closer look on the notion “translation probability” first. From the mathematical point of view, probabilities are just a bunch of nonnegative numbers summing up to 1. For applications in the “real world”, we need to interpret the notion “probability” in a certain way. A popular way to do this is: “The probability p of an event e is the asymptotic relative number of times the event occurs, given a sufficient number of observations (samples)”. This interpretation is sound with the famous Strong Law of Large Numbers, which states that the sample mean converges against the population mean with probability one. Hence, with respect to the translation probability $\Pr(e|s)$ we have to ask the question: “Which (relative) amount of interpreters would translate s as e ?” or alternatively: “How

A human interpreter as
stochastic process

many times (relatively) would one and the same interpreter translate s as e ?⁶ This way, it makes now sense to assign non-negative probabilities to sentences like e_2 : Given a significant number of translators / translations, we can reasonably assume that one of them adds “please” on his own.⁷

We will provide a theoretical framework for statistical methods in machine translation. First of all, we assume that the author of the document containing s had a concept $c \in \mathcal{C}$ in mind when he “emitted” the utterance s . Here, \mathcal{C} is the set⁸ of all concepts one can think of. As we shall see, it is not necessary to provide an unambiguous and complete method to represent these concepts (like an interlingua). It is sufficient to assume that such well-defined concepts exist. So as we try to translate our source sentence s , we first try to extract the concept c from s , the common meaning shared by s and (as we require) e . However, natural languages contain a high amount of ambiguity, the best thing we can do is to guess the possible concepts, providing a probability distribution over the concept space \mathcal{C} . Note that we don’t assume translation to be a deterministic process any more.

Both interpretation as well as generation of a sentence is an indeterministic process

Formally, instead of a well-defined interpretation function $\mathcal{I} : \mathcal{L} \rightarrow \mathcal{C}$, assigning a unique concept to any utterance from a language L , we now have a set of random variables $\mathcal{X} := \{X_s : \Omega \rightarrow \mathcal{C} \mid s \in \mathcal{S}\}$, one for each utterance s . Each X_s maps a probability space Ω onto the family of concepts. Having a probability measure $\Pr : \Omega \rightarrow [0, 1]$ defined over Ω we can therefore construct a probability distribution over the family of concepts $\Pr^{X_s}(c)$. \mathcal{I} assigns each s to its random variable

$$\mathcal{I} : \mathcal{L} \rightarrow \mathcal{X}, \quad s \mapsto X_s \quad (1.4)$$

and analogously

$$\mathcal{G} : \mathcal{C} \rightarrow \mathcal{Y}, \quad c \mapsto Y_c \quad (1.5)$$

We can therefore define

$$\Pr(c|s) := \Pr^{X(s)}(c)$$

having

$$\Pr(e|s) = \Pr(c|s) \cdot \Pr(e|c, s)$$

We assume that e is generated directly out of c , independently of s (note that the concept c contains all the information we need to carry over the meaning of s into the target language), hence $\Pr(e|c, s) = \Pr(e|c) := \Pr^{G(c)}(e)$.

$\Pr(c|s)$ and $\Pr(e|c)$ are rather of theoretical nature. They establish a link between the classical transfer model and the fundamental theorem of machine translation. In real world applications, $\Pr(e|s)$ is virtually always considered directly.

Most statistical MT systems don’t use an interlingua to express intermediate results and perform a direct translation instead

⁶These are two reasonable, but in detail different interpretations of the notion “translation probability”, the first one assuming an inherent model for any Spanish-English translation, the second one modeling each interpreter individually.

⁷Another, more technical reason not to assign zero probability to sentence e_2 is given in section 1.3.5.

⁸For the sake of convenience we consider \mathcal{C} to be a set here, so we can define distributions over it.

1.3 Breaking down the models

Given the statistical framework in the previous section, we face three problems

Decoding We have to perform a search for the target sentence maximizing the a-posteriori-probability, as denoted by the $\arg \max$ operator in equation 1.3. The decoder usually calculates $\Pr(e|s)$ explicitly, so we also have to solve the problem of

Evaluation We need to calculate $\Pr(e|s)$, that is, we want to explicitly map a source and target sentence onto a real number indicating the translation probability.

Training Parameter estimation for the probabilistic models must be possible within a reasonable amount of time.

We start with the second and third problem, as the implementation of the translation model is prerequisite for the decoding process. An overview of a modern SMT decoder is given in section 1.4.



Claude Elwood Shannon, 1916–2001. His work is considered to be fundamental for modern information and communication theory. He also laid the formal foundations of cryptography as a science [Sha49].

Translation as noise removal task

1.3.1 The noisy channel

$\Pr(e|s)$ is virtually never calculated directly. Actually, a significant part of the art of statistical machine translation is to rewrite formula 1.3. A classical way to do this is motivated by Shannon’s model of the noisy channel [Sha48]:

$$\begin{aligned} \arg \max_e \Pr(e|s) &= \arg \max_e \Pr(e|s) \cdot \Pr(s) \\ &= \arg \max_e \Pr(e, s) = \arg \max_e \Pr(e) \cdot \Pr(s|e) \end{aligned} \quad (1.6)$$

The first equation follows from the fact that $\Pr(s)$ is a constant with respect to the $\arg \max$ operator. The second and third equations follow directly from the definition of conditional probabilities.⁹

An ostensive interpretation of formula 1.6 is as follows: A stochastic source emitted a sentence e in the *target* language (denoted by $\Pr(e)$), which then enters a stochastic channel. The noise in the channel now causes the target sentence to be shifted indeterministically into a sentence s in the source language (denoted by $\Pr(s|e)$), which leaves the channel and is observable by the decoder. It is now the job of the decoder to get rid of the noise and reconstruct the (most probable) original sentence \hat{e} .

Note that in figure 1.3, “Have a nice day” might be more probable than “Hello” *a priori*, however, “Have a nice day” is rather unlikely to result in “Hola” at the end of the noisy channel. On the other hand, “Hi there” might most likely result in “Hola”, but “Hello” is more likely *a priori*.

⁹Another way to prove this is using Bayes’ theorem

$$\Pr(e|s) = \frac{\Pr(s|e) \cdot \Pr(e)}{\sum_e \Pr(s|e) \cdot \Pr(e)} \quad (1.7)$$

Note that the denominator simplifies to $\Pr(s)$, being expressed through $\Pr(s|e)$ and $\Pr(e)$, which is the reason why some textbooks use the term “Bayes’ theorem” for a lightweight version of formula 1.7 having only $\Pr(s)$ in the denominator. This is, however, inconsistent with mathematical literature.

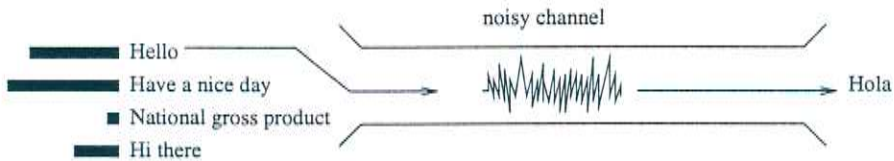


Figure 1.3: Model of the *noisy channel*: We cannot observe the target phrase e that entered the noisy channel. However, we can observe the source phrase s coming out of the channel, and together with the language model probabilities of any target phrase that could have possibly been emitted, weighted by the conditional probability modeling the noise, we can derive the joint probability of any pair e and s .

Some scientists switch the notions “source” and “target” in this context, as the target language of the task is the source language of the channel and vice versa. However, this (ab)use of technical terms is rather confusing and of absolutely no scientific value.

What did we win by this step? The noisy channel approach introduces a new model $\Pr(e)$, the *language model*, which needs implementation and training. Even worse, it does not get rid of the translation model completely, it introduces a *reverse translation model* instead! So why bother with this step at all? The answer is that we can (and usually do) benefit from synergy effects when splitting the translation model into two: The target-to-source translation model measures whether the target sentence actually contains translations for the various concepts expressed in the source sentence (and only for them) while the target language model — among other things — can focus on modeling whether the target sentence is a valid sentence at all (that is, whether the target sentence is grammatical with respect to the target language). A source-to-target translation model would have to model both aspects of translation quality at once. Furthermore, we can use translation models and language models in different levels of the decoding process, as we shall see later.

1.3.2 Translation model

Theoretically, we could implement a translation model by a huge table, storing the translation probability of any source / target sentence pair directly. This is infeasible, of course, both from the aspect of memory space as well as from the aspect of training complexity.

Brown et al. [BPPM93] proposed five models (called IBM-1 through IBM-5) of increasing complexity, introducing the basic notion of *alignments*: Alignments provide a formal specification of word-to-word correspondences for a given sentence pair. This approach eventually leads us to word-to-word translation probabilities, which can actually be stored in a huge table using sparse array techniques.¹⁰

word alignments

Break down TM to the word level

It is important to realize that not all alignments are equally probable (although stated this way in the simplest model IBM-1). Above all, alignments tend to be

¹⁰Such a table is widely known as *lexicon*, although the term “dictionary” would probably have been a better choice here.

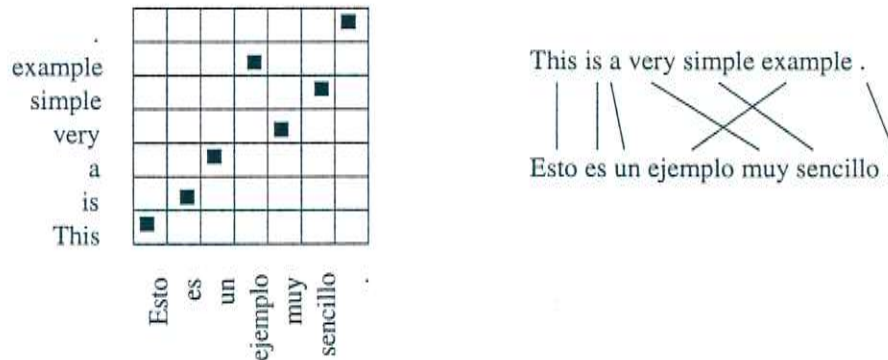


Figure 1.4: A sample of an *alignment* between source and target words of a sentence pair. A black box in the grid means that the source word is aligned to the target word. There is one nonmonotonicity caused by the fact that the substantive “example” follows its attributes in English, while the attributes follow the substantive in Spanish.

monotone, that is, alignment diagrams like fig. 1.4 tend to have boxes on the main diagonal.

The alignment shown in fig. 1.4 is also the most probable one, called *Viterbi alignment*. In fact, in the example shown, the Viterbi alignment is the *only* alignment that really makes sense, so we can assume an *alignment probability* close to one: $\Pr(a|s, e) \approx 1$.

The *Viterbi alignment* often claims most of the probability mass

[BPPM93] not only introduced ways to break down a sentence-to-sentence translation model to the word level, they also provided methods to efficiently train the model parameters. In general, they use a two step algorithm widely known as *Expectation-Maximization (EM)* using alignments as a hidden variable.

Vogel et al. [VNT96] introduced a modified version of the IBM-2 model called *HMM*, where alignment positions are conditioned on the alignment position of the previous word. The HMM model performs quite well if we want to allow some large jumps while assuming that alignments stay locally monotone.

An overview about further improved word alignment models has been provided by Och and Ney [ON00], including alignments to the empty word or alignments that depend on the part-of-speech tag¹¹ of the source resp. target word.

Zhao and Vogel [ZV03] published an interesting alternative statistical approach called *bilingual bracketing*, which is based on hierarchical reordering trees.

1.3.3 Language model

Language models model the probability that a certain sentence or segment is being emitted by an indeterministic language source. What does that mean? In the first place, it seems to be reasonable to assume

$$\Pr(\text{“Time flies like an arrow”}) \gg \Pr(\text{“Time in flies have”})$$

¹¹A grammatical tag indicating the word class, e. g. “noun”, “verbal phrase”, information about numerus and/or genus etc.

for an English language model. However, language models do not just provide a figure of merit of grammaticity (just as translation models are not just figures of merit of translation quality, see p. 4).

One could argue that “Time in flies have” cannot be emitted by a stochastic English source at all (and should therefore have a probability of zero). On the other hand, note that an English native speaker could emit these words at any time, which contradicts the assumption of “Time in flies have” being an “impossible” event. The question is, whether a native speaker that emits “impossible” respectively incorrect sentences can still be considered a stochastic source of English language in that moment. Consequently, no one can forbid an English native speaker to emit a French sentence or even just noise¹².

Apart from the ideological motivation, there are some straightforward technical reasons not to assign zero probability to *any* sequence of words; a general technique to achieve this is *smoothing* (see section 1.3.5).

Language models are heavily used in modern Automatic Speech Recognition (ASR) systems, and are therefore well-known to the scientific community. There exists a broad variety of methods to model the probability of a sentence. The by far most popular approach is based on *n-grams*, that is, we model the probability of a sequence of words by the product of the probabilities of the words given the $n - 1$ preceding words. For bigrams, we have:

$$\begin{aligned} \Pr(e_1 \dots e_k) &= \prod_{j=1}^k \Pr(e_j | e_1 \dots e_{j-1}) \\ &\approx \prod \Pr(e_j | e_{j-1}) \end{aligned} \quad (1.8)$$

These n-gram probabilities are quite easy to estimate:

$$\Pr(e_j | e_{j-1}) \approx \frac{\#(e_{j-1}e_j)}{\#(e_{j-1})} \quad (1.9)$$

where $\#(e)$ denotes the *word count* of e , the total number of times the event e occurred (in this case, the total number of times the bigram $e_{j-1}e_j$ respectively the word e_{j-1} occurred in the training corpus).

1.3.4 Other models

Translation and language are not the only stochastic processes one can model. Both translation and language model technically require a *sentence length model* when breaking the model down to the word level.

Furthermore, it is important to note that the noisy channel approach (see (1.3.1)) is one, but not the only way to break down equation (1.3). An alternative approach that allows to incorporate various models more easily is the *Maximum Entropy Approach* [BPP96]:

$$\Pr(e|s) = \frac{\exp \left[\sum_{m=1}^M \lambda_m h_m(e, s) \right]}{\sum_{\tilde{e}} \exp \left[\sum_{m=1}^M \lambda_m h_m(\tilde{e}, s) \right]} \quad (1.10)$$

The maximum entropy approach generalizes the noisy channel approach from eq. (1.6)

¹²Models for noise are actually part of any up-to-date speech recognition system.

The $h_m, m = 1, \dots, M$ are a set of *feature functions*, which are weighted by a corresponding set of model parameters $\lambda_m, m = 1, \dots, M$ that need to be trained [CF04]. The maximum entropy approach leads to the following decision rule:

$$\hat{e} = \arg \max_e \Pr(e|s) = \arg \max_e \left\{ \sum_{m=1}^M \lambda_m h_m(e, s) \right\} \quad (1.11)$$

1.3.5 Smoothing

Typical problems in connection with statistical models are

Unseen events due to data sparseness **Unseen events** Recall the n -gram language model from page 9. For a language consisting of 100,000 different words (a rather moderate size), there are 1,000,000,000,000,000 possible trigrams — a vast number that cannot be covered by any existing corpus. Even if only a small fraction of these trigrams are actually “valid”, it is impossible to observe them all during training,¹³ which means there are non-impossible events that have no statistical evidence in the training data. The situation gets even worse for larger history lengths.

Similar arguments hold for translation models, among others. Events erroneously assigned zero probability can lead to the loss of valid solutions of eq. (1.3).

Overfitting As a direct consequence of the data sparseness problem, due to the latent lack of statistical evidence, statistical models tend to focus too much on the training data, often basing parameter estimation on events that have only been seen once (“*singleton*”) or twice. This disability to generalize beyond training data is known as *overfitting*.

Complex search space Even if we have perfect models, the search space for eq. (1.3) is full of local minima and zero-probability-plateaus. Such a search space is hard to search, even with sophisticated search algorithms (see fig. 1.5).

Model smoothing is important! A general approach to cope with all three problems at once is called *smoothing*. The usual way to do this is to interpolate a high-order model with a more general one.

1.4 Decoding

The *decoder* is the part of the SMT system that deals with the search problem as indicated by the $\arg \max$ operator in eq. (1.3).

Most contemporary decoders rely on *translation lattices*:

¹³“There is no data than more data!” — the fact that there is virtually never enough data to cover all possible events for training is referred to as “data sparseness”.

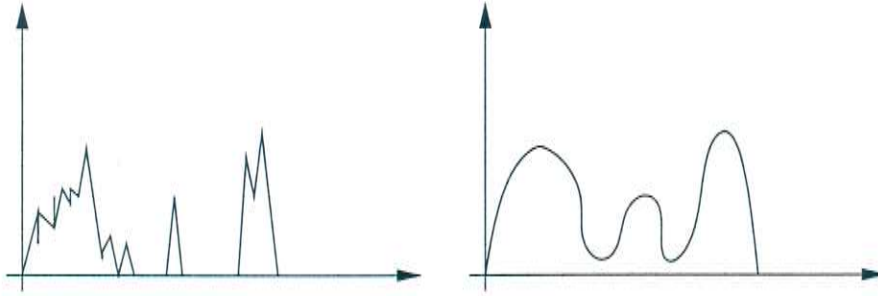


Figure 1.5: Due to *smoothing* of the (1-dimensional) search space (**Right**), the global maximum is much easier to find.

Definition 1.1 (Translation lattices)

A translation lattice L is an acyclic graph $(\mathcal{N}, \mathcal{E}, n_0, \mathcal{F})$ with a finite set of nodes \mathcal{N} , a finite set of annotated edges $\mathcal{E} \subset \mathcal{N}^2 \times \mathcal{V}_L^* \times \Omega$ (where $\mathcal{V}_L^* = \cup_n \mathcal{V}_L^n$ is the set of all finite (target) phrases over a finite lattice target vocabulary \mathcal{V}_L and Ω is the score space), an initial node n_0 and a set $\mathcal{F} \subset \mathcal{N}$ of final nodes.

Translation lattices are sometimes also referred to as *stochastic finite-state transducers* [ABC⁺99, CV04] or *word graphs* [UON02, ZN05].

Usually, we can identify each node n_i , $i = 0 \dots k$ with a position between two words in the source sentence $\mathbf{s} = s_1 \dots s_k$, $\mathcal{F} = \{n_k\}$ and $i < j$ for any edge $E = (n_i, n_j, \mathbf{t}, \omega) \in \mathcal{E}$. In this case, traversing E denotes the act of translating the source phrase $s_{i+1} \dots s_j$ into the target phrase \mathbf{t} .

Earlier approaches include a *stack decoder* by Wang and Waibel [WW97].

Fig. 1.6 depicts a schematic view on a lattice-based decoder. Decoding is carried out in two steps:

1. The first step is a *lattice generation step*. The source sentence is being transformed into a lattice, where each edge translates some segment of the source sentence (“source phrase”) into a target phrase. Some statistical models, above all the translation model are applied by assigning some kind of costs to the edges.
2. In a second step, the decoder searches for an optimal path through the lattice with respect to the model scores (*path search step*). Costs associated to the edges are accumulated and language model scores are applied (among others).

The use of lattices has become popular in domains of Automatic Speech Recognition. However, translation lattices (which are nothing more than modified ordinary automata) can only accept or generate Chomsky-3 [Goo97b] (that is, linear) languages [Goo97a]. However, as opposed to speech recognition tasks, the alignment between source and target sentence is generally not monotone in language translation tasks, which makes pure linear decoding methods inadequate.

Translation lattices only support monotone alignments by nature

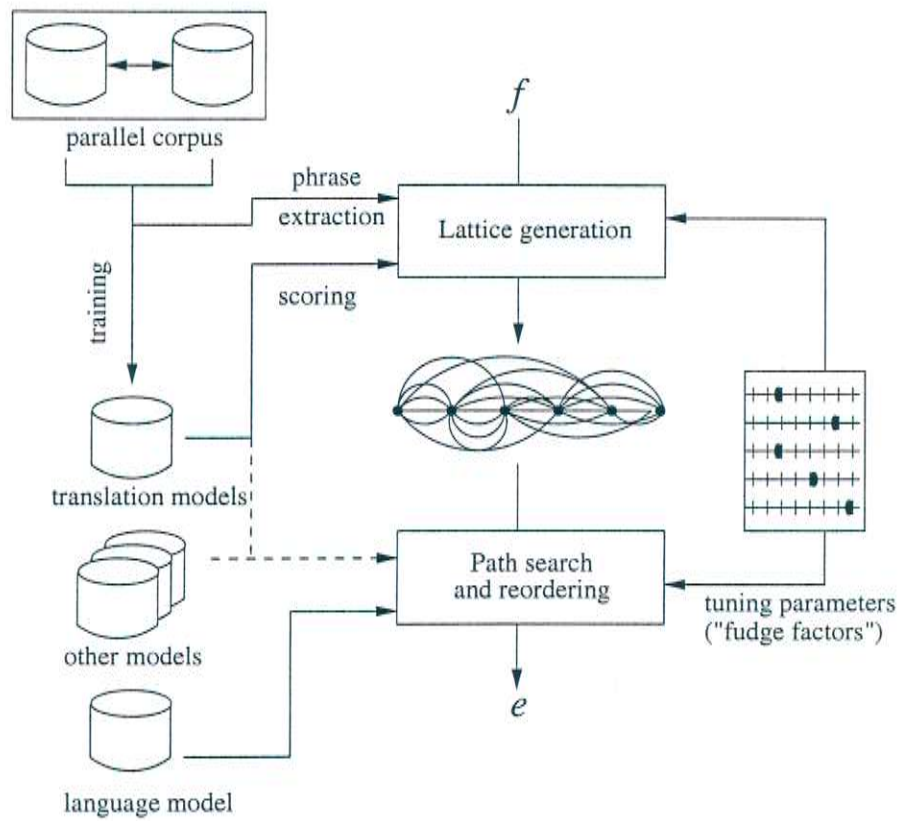


Figure 1.6: A typical decoder, based on a noisy channel model and translation lattices. "Other models" could be statistical sentence length models, phrase segmentation models etc.

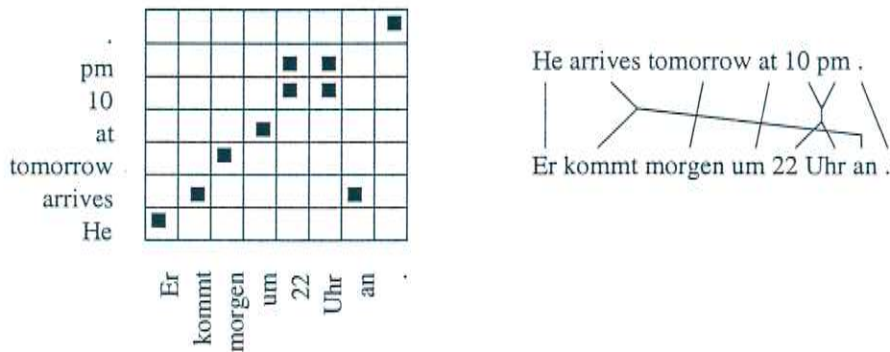


Figure 1.7: German compound verbs caused a major nonmonotonicity in this example: The English verb “arrive” is translated by the German verb “ankommen”, which is splitted into its stem “kommen” and the particle “an”. The particle “an” could be arbitrarily far away from its stem, which makes global reordering techniques necessary — a major difficulty also for human interpreters.

The notion *reordering* denotes the task of coping with nonmonotonicities between source and target sentences. Reordering comes in two flavors:

Local reordering Caused by a short-range nonmonotonicity, local reordering usually only involves the rearrangement of two or three words in a very local context. In Spanish, for example, attributes usually follow its noun while they are prepended in English (see fig. 1.4). Local reordering can be handled surprisingly well by phrase-based translation methods.

Global reordering In some language pairs, words have to be rearranged over a larger context: (partial) translations of a word at the beginning of the source sentence can be found at the very end of the target sentence and vice versa. Such long-range nonmonotonicities are hard to resolve, however, they typically only occur for certain languages respectively language pairs (including English-German, see fig. 1.7).

A common way to handle reordering in a lattice-based decoder is to make the decoder reorder the edges of a path during or after the path search step in order to improve the model score (by increasing the language model score and/or by gaining on a reordering model score). This effectively means a phrase-wise reordering of the words of the source sentence¹⁴. As there are $n!$ possible arrangements for a path consisting of n edges, pruning becomes necessary to avoid a combinatorial explosion of the search space.

Alternatively, it is possible to encode the reordering state directly in the word graph [KVM⁺05, ZON02].

¹⁴An alternate approach would be to reorder the target words of a monotone decoding result in a post-processing step.

1.5 Evaluation

In order to evaluate the quality of a translation system, one could just ask a human native speaker of the target language to do so. Given the system output on a given set of test sentences and a corresponding set of high-quality references¹⁵, the human judge could, for instance, assign a number between 1 and 10 to each test translation or even multiple scores for various aspects of translation quality. Such a way of evaluation is, however, highly subjective, slow, and expensive as it requires human interaction.

Manual resp. human evaluation is slow and expensive

Papineni et al. [PRWZ02] proposed a metric called BLEU, which compares a set of test translations against a set of reference translations. BLEU supports multiple references in order to cope with the ambiguity of the translation process. As opposed to the Word Error Rate (WER), a popular figure of similarity used in Automatic Speech Recognition, BLEU allows a limited amount of nonlinearity between translation hypothesis and reference sentence.

BLEU counts the number of hypothesis n -grams that occur both in the candidate translation as well as in one of the references, where each reference n -gram can only be covered once. BLEU then calculates the geometric mean $\bar{h} = \sqrt[N]{\prod_{n=1}^N h_n}$ of the various hit rates¹⁶ (that is, the number of matching n -grams divided by the total number of n -grams in the hypothesis) for varying n -gram size n . To make sure that the hypothesis string contains enough events (words), the mean hit rate is multiplied with a non-linear *brevity penalty*, which is 1 if the candidate translations are longer than the corresponding best-match references and exponentially decaying in the ratio between effective reference corpus length and test corpus length otherwise.

BLEU has been proven to be highly correlated with human judgment.

An alternative, extended approach has been provided by the NIST consortium [NIS].

Melamed et al. [IDM03] used a modification of precision and recall (see 3.2.1) that also supports multiple references to show that standard measures can outperform the BLEU metric in the matter of reliability.

1.6 System overview

The Statistical Translation Toolkit (STTK) [V⁺] is being developed by InterACT research¹⁷. The STTK provides a basic development framework, a decoder and training procedures for various models.

The SRI toolkit [TL] was used to train and implement language models.

¹⁵Instead of references, one could directly consult a human interpreter with knowledge of both source and target language here. However, monolingual staff is usually cheaper and easier to acquire.

¹⁶Note that the mean hit rate becomes 0 if one of the h_n becomes 0.

¹⁷InterACT research is a joint research lab at Carnegie Mellon University, Pittsburgh, USA and the University of Karlsruhe (TH), Germany. See <http://interact.ira.uka.de>.

Corpus:	EPPS		UN	
Languages:	English	Spanish	English	Spanish
Segments:	1,162,309		618,002	
Words:	30,507,604	31,948,500	18,267,894	21,047,570
Words per segment:	26.2	27.5	29.6	34.1
Vocabulary size:	86,790	123,148	79,964	100,748
Singletons:	32,429	42,636	33,858	39,229

Corpus:	ACL	
Languages:	English	French
Segments:	688,031	
Words:	13,807,006	15,590,687
Words per segment:	20.1	22.7
Vocabulary size:	61,619	80,495
Singletons:	22,971	28,990

Table 1.2: Training corpora

Training corpora (table 1.2) are based on the minutes of the European Parliament Plenary Sessions (EPPS) which were provided in the course of the TC-STAR [TCS] project; the United Nations Parallel Texts corpus based on documents of the Office of Conference Services at the UN in New York, provided by the Linguistic Data Consortium (LDC) [LDC]. We used English and Spanish parallel data from both corpora, furthermore, we used additional English-French parallel data also based on the EPPS corpus which was provided by Phillip Köhn [Köh].

For decoding, we used PESA online phrase extraction and IBM1 forward and backward lexica (word-to-word translation probabilities). All language models were trained on the corresponding EPPS trainsets. Decoding parameters had been manually tuned on a separate development set of 1,000 sentences respectively.

For the English-Spanish data tracks, the Final Text Edition (FTE) development sets from the first TC-STAR evaluation run for Spoken Language Translation served as test sets in both directions. As some metrics proposed in this thesis do not support multiple references (yet), and the TC-STAR development set provides two references, only one reference (the first of the two) was used. A English-French test set was separated from the training data. See table 1.3 for details.

Troughout this thesis, we used BLEU (see sec. 1.5) with a single reference and a maximal n -gram size of $N = 4$ to evaluate candidate translations.

Testset:	tcstar-enes		tcstar-esen	
Languages:	English → Spanish		Spanish → English	
Segments:	1,008		1,008	
Words:	22,812	20,076	23,624	21,828
Vocabulary size:	2,760	3,301	3,440	2,599
Singletons:	1,368	1,931	1,930	1,258
Voc. coverage:				
(EPPS)	2,680 (97%)	3,211 (97%)	3,352 (97%)	2,523 (97%)
(UN)	2,578 (93%)	3,065 (92%)	3,198 (92%)	2,446 (94%)

Testset:	acl-enfr/acl-fren	
Languages:	English ↔ French	
Segments:	2,000	
Words:	57,951	66,200
Vocabulary size:	6,054	7,252
Singletons:	3,043	3,817
Voc. coverage (ACL):	5,921 (97%)	7,114 (98%)

Table 1.3: Test sets

Chapter 2

Oracle Decoding

In Machine Translation, *oracle experiments* denote the act of using a reference translation in order to perform a model-free search for the optimal hypothesis. As a substitute for an ordinary decoder, we may also speak of *oracle decoding*. The need for a reference translation restricts applications of oracle experiments to evaluation and optimization issues. Oracle experiments have been successfully applied to domains of Automatic Speech Recognition, where ASR recognition lattices are being forcefully aligned to reference transcriptions. Given this context, we may also speak of *forced alignments*.

In the course of this thesis, two forced aligners (resp. oracle decoders) finding an optimal path through a translation lattice with respect to a reference translation have been implemented as part of the STTK. They differ in the particular objective function they use as optimization criterion for the path search.

The first one will be referred to as *Levenshtein decoder* (see sec. 2.1.1) and finds a path through a lattice minimizing the *Minimal Edit Distance (MED)* (sometimes also called *Levenshtein distance*, after its inventor) between translation hypothesis and reference translation. Minimal edit distance denotes the minimal number of (word) insertions, substitutions and deletions one needs to transform one sentence into the other¹. While the minimal number of insertions, deletions and substitutions is well-defined, the concrete sequence of manipulations necessary for the transformation is ambiguous in general. The Minimal Edit Distance is closely related to the *Word Error Rate (WER)* (see eq. 18), a metric popular (not only) in Automatic Speech Recognition.

The second forced aligner (see sec. 2.1.2) optimizes for the *Position-independent Minimal Edit Distance (PMED)* between hypothesis translation and reference.

There are various applications for the performance (i. e. BLEU score) of the hypothesis returned by a forced aligner. Note that the evaluation function used to assess the oracle hypothesis² does not necessarily have to be the objective function the forced aligner optimizes for. In particular, hypotheses returned by the Levenshtein decoder are generally not optimal with respect to BLEU,



Vladimir I. Levenshtein, * May 20, 1935, Moscow, USSR [Lev02]. L. introduced the metric which was named after him in 1965 [Ata99].

¹Note that Minimal Edit Distance is symmetric, that is, $MED(x, y) = MED(y, x)$.

²That is, the hypothesis optimal with respect to the *objective function*.

however, we can assume that they are a good approximation for that objective (also see 2.4.2).

If we speak of *oracle score*, we thereby mean — if not differently indicated — the BLEU score (which was used as string-to-string(s) evaluation metric for hypothesis translations throughout this thesis, see p. 14) of the oracle translation.

A well-known oracle metric where the objective function matches the evaluation function is the *Graph Word Error Rate (GWER)* [ZN05], that is, the minimal Word Error Rate (eq. 2.2) of all translations represented by the lattice:

$$\text{GWER}(L, \mathbf{r}) := \min_{p \in \mathfrak{P}(L)} \text{WER}(\mathbf{t}_p, \mathbf{r}) \quad (2.1)$$

where \mathfrak{P} is the set of all paths³ in the translation lattice L , $\mathbf{t}_p = t_1 \cdots t_k$ is the translation represented by the path p and $\mathbf{r} = r_1 \cdots r_m$ is the reference translation corresponding to L . The *Word Error Rate (WER)* between hypothesis and reference translation as used in eq. 2.1 is defined to be

$$\text{WER}(\mathbf{t}, \mathbf{r}) := \frac{\text{MED}(\mathbf{t}, \mathbf{r})}{|\mathbf{t}|} \quad (2.2)$$

where $|\mathbf{t}| = |t_1 \cdots t_k| = k$ is the length of the hypothesis translation. The *Position-Independent Word Error Rate (PER)*⁴ is defined analogously, using the Position-independent Minimal Edit Distance instead of MED in eq. 2.2.

If we evaluate the hypothesis returned by the Levenshtein decoder using its objective function (using the same reference translation \mathbf{r} that was used for the path search), the resulting score is called *Graph Error Rate (GER)* [UON02], which is the smallest Minimal Edit Distance of all paths through a translation lattice:

$$\text{GER}(L, \mathbf{r}) := \min_{p \in \mathfrak{P}(L)} \text{MED}(\mathbf{t}_p, \mathbf{r}) \quad (2.3)$$

Although looking very similar, Graph Error Rate and Graph Word Error Rate are not the same. Above all, hypotheses optimal with respect to MED are generally not optimal with respect to WER and vice versa. Note that, in addition to depending on its Minimal Edit Distance, the Word Error Rate is antiproportional to the length of the candidate translation, which means that GWER has to optimize for both a small Minimal Edit Distance and a long hypothesis translation simultaneously, while GER only regards the Minimal Edit Distance, independently from the length of the hypothesis translation.

We will give an example to demonstrate the difference: Imagine the smallest Minimal Edit Distance of all paths through a lattice (that is, the Graph Error Rate) is 3, and there are three hypotheses (paths) producing a candidate translation having a Minimal Edit Distance of 3 with respect to the reference translation \mathbf{r} : one using three insertions (the candidate translation being three words shorter than \mathbf{r}), one using three deletions (the candidate translation being three words longer than \mathbf{r}) and one using three substitutions (the candidate translation being as long as \mathbf{r}). All three hypotheses result in the same Graph

³If we want to take *reordering* (see p. 11) into account, \mathfrak{P} must be the set of all permutations of all paths through L .

⁴The correct abbreviation would probably be “POWER”, but in literature, one will usually find the abbreviation used here.

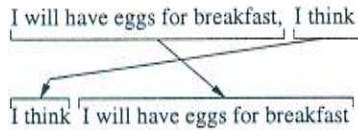


Figure 2.1: These highly similar sentences lead to a comparatively high Levenshtein distance of 4 (not counting the comma) due to the phrase “I think” that has been moved to the front (two insertions and two deletions are necessary).

Error Rate, however, GWER would pick the one using three deletions, as this would maximize the denominator in eq. 2.2 (or any other hypothesis that produces an even longer candidate translation, even if it produces more errors,⁵ as long as the quotient in eq. 2.2 becomes minimal). However, we can force the Minimal Edit Distance to at least approximately simulate the behavior of the Word Error Rate (at least with respect to hypothesis selection) by setting the cost for a deletion slightly lower and the cost for an insertion slightly higher than the cost for a substitution (which is normally 1). This way, we enforce the Minimal Edit Distance, and therefore also the Graph Error Rate (resp. the Levenshtein decoder), to prefer longer translations, if there is a choice.⁶ Using this trick, we can use the Levenshtein decoder in order to compute an approximation of the Graph Word Error Rate.

An oracle decoder objecting directly for an optimal BLEU score has been implemented recently by Zens et al. [ZN05]. The corresponding oracle score is then called *GBLEU*, which is an upper bound for the BLEU score any path search could achieve on the particular lattice. Zens et al. also implemented an oracle decoder calculating *GPER*, the smallest Position-independent Word Error Rate of all candidate translations possibly emitted by the translation lattice.

2.1 Implementation

As opposed to BLEU, Minimal Edit Distance cannot handle larger nonmonotonocities very well (fig. 2.1). For this reason, in addition to the Levenshtein decoder, a second forced aligner has been implemented, objecting to the *Position-independent Minimal Edit Distance (PMED)*. The PMED is robust against nonmonotonocities as word order is irrelevant to this metric. However, PMED naturally does not account for fluency⁷ and the direct correlation between PMED

MED penalizes non-monotone alignments, esp. long-range reorderings

⁵Well, not exactly in this scenario. Note that for a given number of errors n , the Word Error Rate is bounded by $\frac{n}{|r|+n}$ and 1, because — as indicated before — the best that could happen is that all n errors are deletion errors, maximizing the length of the candidate translation (which is $|r| + n$ in this case). However, $\frac{n}{|r|+n}$ is a strictly monotonically increasing function with $\frac{n}{|r|+n} \nearrow 1$, so that any hypothesis producing a candidate translation that causes more than three errors would have a larger Word Error Rate than the $\frac{3}{|r|+3}$ of the hypothesis mentioned in the example.

⁶In the course of this thesis, however, we configured the Levenshtein decoder to set the cost of an substitution to 0.99, while both the cost of insertions and deletions was set to 1, in order to favor hypotheses that produce a candidate translation that has a length close to the length of the corresponding reference translation.

⁷Here: measure of how grammatical and well-formed a sentence is

		4	3	3	3	2	2
Madrid		3	2	2	2	1	2
to		2	1	1	1	2	3
went		1	0	1	2	3	4
I		0	1	2	3	4	5
		I	never	went	to	Paris	

Figure 2.2: The Dynamic Programming matrix used to calculate the Minimal Edit Distance between “I went to Madrid” and “I never went to Paris”. Note that the Levenshtein metric is symmetric, but we will usually assign the reference sentence to the columns.

and BLEU is rather loose (optimal hypotheses with respect to PMED tend to be a bag of words of the reference sentence in arbitrary order).

2.1.1 The Levenshtein decoder

Efficient calculation of the Levenshtein distance between two word sequences is a classical application of *Dynamic Programming (DP)*: A table or matrix storing partial results or costs is systematically filled by the DP algorithm until the total result can be calculated (fig. 2.2).

Each row $i = 0 \dots l$ corresponds to a position between two hypothesis words⁸ (where l is the length of the hypothesis in words), each column $j = 0 \dots m$ to a position between two reference words. Each entry (i, j) in the table denotes the Minimal Edit Distance between the first i hypothesis words and the first j reference words. The total edit distance is therefore the entry in the upper right corner.

The values of the first row / first column (indicating an empty hypothesis respectively an empty reference sentence) are always $0, 1, 2, 3, \dots$. Furthermore, we can calculate any entry in cell (i, j) if we know the three values of the cells to the left, below, and diagonally to the lower left (fig. 2.3). We then distinguish three cases, choosing the scenario *minimizing* the cost for cell (i, j) :

- Move into cell (i, j) from the cell to the left. This denotes the act of skipping a reference word, therefore committing an *insertion error* (we will have to insert the skipped reference word later). The value of cell (i, j) is then the value of cell $(i, j - 1)$ plus one for the extra error.
- Move into cell (i, j) from the cell below. This denotes the act of skipping (deleting) a hypothesis word, which indicates a *deletion error*. Similarly

⁸Position 0 is in front of the first word, position l corresponds to the sentence end, behind the last word of the sentence.

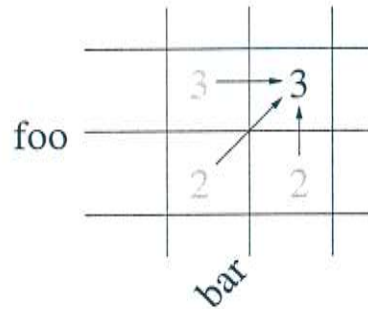


Figure 2.3: Step in the Dynamic Programming algorithm calculating the Levenshtein distance between two sentences. In this example we can freely choose between a substitution of “foo” by “bar” or a deletion of “foo”, both leading to a partial edit distance of 3. An insertion of “bar” would lead to a higher cost of 4, this option is therefore discarded.

to the first case, the value of cell (i, j) is the value of cell $(i - 1, j)$ plus one.

- The third case must be distinguished into two sub-cases: Moving from cell $(i - 1, j - 1)$ into cell (i, j) denotes the act of replacing reference word j by hypothesis word i . If both words mismatch, this indicates a *substitution error* and the value of cell (i, j) is the value of cell $(i - 1, j - 1)$ plus one⁹ for the extra error. Otherwise, if both words are equal, no error occurs and the value of cell $(i - 1, j - 1)$ can be carried over into cell (i, j) without any extra cost.

For each cell, *backpointers* store information which previous cell (to the left, to the bottom left or below) led to the actual value of the cell. We can therefore backtrack through the trellis from the final cell in the last column / row in order to get a path through the trellis (*Levenshtein path*) indicating the concrete edit operations necessary to transform the hypothesis sentence into the reference sentence. As opposed to the Levenshtein distance itself, the Levenshtein path is ambiguous: generally, there are various possible ways to transform the hypothesis sentence into the reference sentence using the same minimal number of edit operations.

During decoding, the complete hypothesis sentence is not known. However, the STTK decoder expands partial hypotheses monotonously with respect to the target sentence, hence for any partial hypothesis, the first i words of the final translation hypothesis are already known. Consequently, this implies that the first $i + 1$ rows of the DP matrix for the calculation of the edit distance are already known (fig. 2.4). We can therefore assign a partial DP matrix to any partial hypothesis; as the values of the remaining cells of any¹⁰ resulting DP matrix depend on and only on the values of the last (known) row (due to

⁹As mentioned before (p. 19), the cost of a substitution was set 0.99 for all experiments, in order to favor oracle translations that have a length close to the length of the reference translation.

¹⁰Note that the eventual (complete) DP matrix is not unique. The various ways to complete

shop	5	4	3	3	2	1	store	5	4	3	3	2	2
shoe	4	3	2	2	1	2	shoe	4	3	2	2	1	2
a	3	2	1	1	2	3	a	3	2	1	1	2	3
is	2	1	0	1	2	3	is	2	1	0	1	2	3
Where	1	0	1	2	3	4	Where	1	0	1	2	3	4
	0	1	2	3	4	5		0	1	2	3	4	5
	Where	is	the	shoe	shop		Where	is	the	shoe	shop		

Figure 2.4: Two different expansions of the partial translation hypothesis “Where is a shoe ...” against the *same* reference “Where is the shoe shop”. Note that the first five rows of both DP matrices are identical.

the nature of the DP algorithm), it is sufficient to store only that row for any (partial) hypothesis of the Levenshtein forced aligner.

The design of the Levenshtein forced aligner was based on a STTK decoder class. Fig. 2.5 depicts an UML ([BRJ99]) diagram of the corresponding hypothesis class.

The state of a (partial) hypothesis object is defined by the last line of the corresponding (partial) DP matrix (as discussed before), backpointer information and a lattice state. Above all, the lattice state indicates which source words have already been translated. More generally, two hypotheses are in the same lattice state if and only if their search spaces (that is, the paths through the lattice that can still be expanded) are identical. For a monotone decoder, the lattice states are isomorphic to the lattice nodes, but for decoders that implement a reordering strategy (see 1.4), lattice states can be more complicated. A hypothesis is *complete* (or *final*) iff all the source words have been translated.

Partial (i. e. incomplete) hypotheses are expanded over lattice edges, leading to a new hypothesis. Hypothesis expansion corresponds to the translation of a single phrase of the source sentence. The decision over which edges a partial hypothesis is to be expanded depends on the decoder’s reordering and pruning strategies.

Hypothesis recombination

In order to reduce the complexity of the search, the expansion step is followed by a second step, the *recombination step*. The purpose of the recombination step is to prune away hypotheses for which it is already evident that they will eventually be outperformed by another hypothesis. In order to be comparable,

the partial hypothesis generally lead to different (complete) DP matrices. Even the matrix size (that is, the number of rows) depends on the further expansion of the partial hypothesis (note that the number of columns is determined by the (fixed) hypothesis sentence).

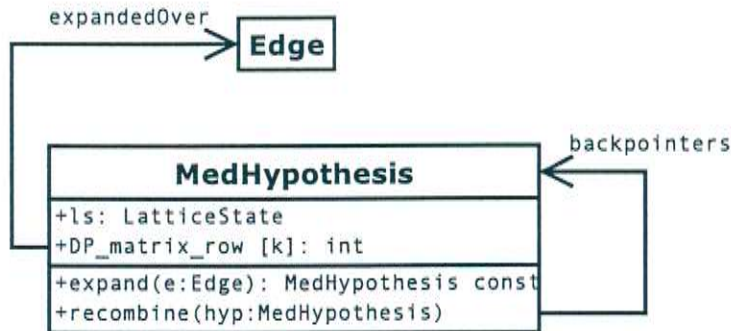


Figure 2.5: A simplified UML diagram of the (partial) hypothesis class of the Levenshtein forced aligner for a reference of $k - 1$ words. The length of the reference sentence is indicated by the size k of the `DP_matrix_row` vector. The *initial hypothesis* is the only hypothesis that has not been `expandedOver` a lattice edge.

two hypotheses must match the same lattice state (so they will see the same events in the future) as well as any other state impacting the final score (e. g. a language model state — this is not relevant for a model-free forced aligner, though).

Fig. 2.6 gives an example of the scenario: A hypothesis is being expanded over an edge which apparently emits four target words (as indicated by the difference of the first entry of the DP matrix row `DP_matrix_row`: The first component of the `DP_matrix_row` vector always indicates the number of *deletions* necessary to align the current hypothesis to an empty reference sentence — and therefore the total number of target words emitted by the hypothesis so far). The expansion of the matrix row over the four new target words leads to a new `DP_matrix_row` vector and the expansion of the edge itself to a new lattice state¹¹.

For that given lattice state, let `bestHyp` be the currently “best” hypothesis (whatever this means for now) having that lattice state. How can we now recombine these two hypotheses? For an ordinary decoder (as opposed to forced aligners), we would just prune away the hypothesis h^- with the lower partial score s^- and keep the better hypothesis h^+ having a higher score s^+ . We can do this, as any sequence of expanded edges (e_i) leading to a total score $t^- = s^- + r$ for h^- leads to an even higher score $t^+ = s^+ + r = t^- + \underbrace{s^+ - s^-}_{>0}$. (The fact

that (e_i) is also a valid sequence of edges to be expanded for h^+ follows from the assumed comparability (i. e. same lattice state); r being the same for both h^+ and h^- follows from the fact that the decoder (at least the STTK decoder) accumulates scores linearly.) However, we don’t have a single “partial score” for a partial hypothesis of the Levenshtein forced aligner. Instead, we have a whole vector of partial edit distances. Technically there are two ways to manage this:

¹¹For the sake of simplicity, the lattice state is indicated by an integer in our example, which is only sufficient for monotone decoding.

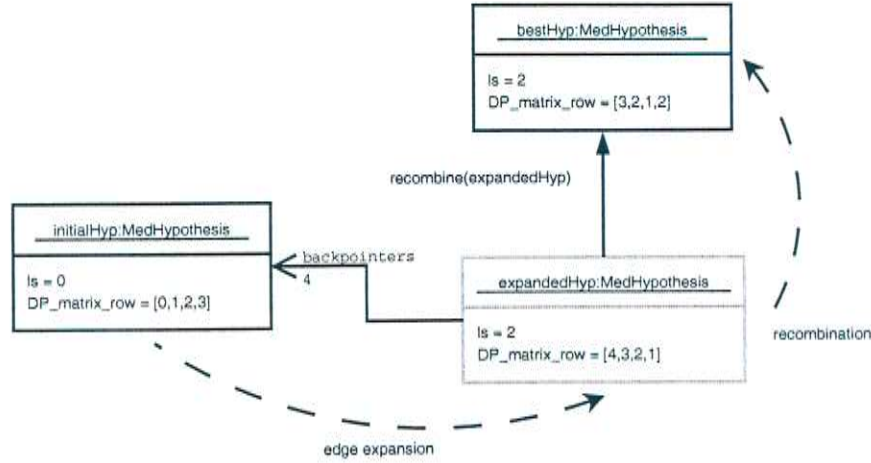


Figure 2.6: The two steps of hypothesis expansion: edge expansion and hypothesis recombination. In this case, the partial hypothesis being expanded is the *initial hypothesis* of no (source) words being translated, no edges being expanded so far.

- Recombine partial hypotheses with matching lattice states if and only if all edit distances in `DP_matrix_row` match as well. In this case, we can freely choose between the two candidates, as they only differ by their backpointer information: They will eventually perform equally with respect to edit distance, due to the identical DP matrix rows.

However, the prerequisite of the `DP_matrix_row` vectors having to be identical limits the extent by which recombination can occur. In practice, the remaining search space is still too big to be searched entirely, pruning methods become necessary.

- Instead of considering the entire vector of `DP_matrix_row` as a state, compare the entries of the DP matrix row individually, as a set of partial scores. Out of all hypotheses sharing the same lattice state, construct a new "best" hypothesis which, as j th entry in its `DP_matrix_row`, has the lowest available j th entry of all considered `DP_matrix_row` vectors:

$$\text{bestHyp.DP_matrix_row}[j] := \min \{ \text{hyp.DP_matrix_row}[j] : \text{hyp.ls} = s_0 \} \quad (2.4)$$

This best hypothesis can be constructed iteratively.

In order for this to work, we have to introduce individual backtracking information (back to the hypothesis that had originally been expanded as well as the lattice edge over which the hypothesis had been expanded) for each entry of the `DP_matrix_row` vector. The (expanded, not yet recombined) hypothesis that contributed the lowest partial edit distance for the j th entry of `bestHyp.DP_matrix_row` also provides the corresponding hypothesis and edge backpointers.

The second approach is preferable, as it reduces the complexity of the search

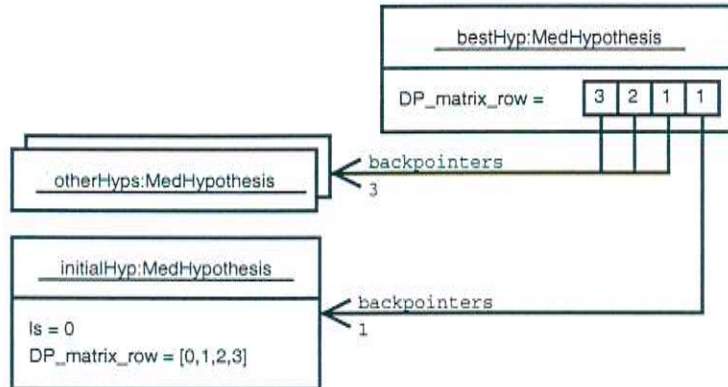


Figure 2.7: After the recombination (fig. 2.6), one of the backpointers actually points on the hypothesis that had been expanded in the first place.

space sufficiently to make a complete search without pruning possible¹². An example of the recombination result of the scenario given in fig. 2.6 is shown in fig. 2.7.

However, the proof that this fine-grain recombination strategy is by all means “valid” is highly non-trivial. We have to verify that two properties of the search are not violated:

Completeness The search finds the optimal solution, that is, no final hypothesis h in the search space \mathcal{H} indicated by the partial hypothesis h_p and lattice L has a Minimal Edit Distance better than the optimal Minimal Edit Distance \hat{d} returned by the search algorithm.

$$\hat{d} \leq \{\text{MED}(h, \mathbf{r}) : h \in \mathcal{H}(h_p, L)\} \quad (2.5)$$

We have to make sure the search does not “lose” optimal solutions.

Soundness The total Minimal Edit Distance of the optimal final hypothesis \hat{h} , as indicated by the last entry of the `bestHyp.DP_matrix_row` vector, matches the true Minimal Edit Distance between the translation hypothesis (as constructed over the backpointers of the final hypothesis) and the (fixed) reference sentence.

It is important to note that using recombination this way, the vector `DP_matrix_row` does not contain a row of an actual DP matrix any more; it rather recombines the most promising cells of many DP matrices. It is not given that the resulting modified DP algorithm is still correct: As the resulting edit distance d is calculated by means of a mix of many DP matrices (one for each translation hypothesis), the corresponding translation hypothesis might be a “mix” of many translation hypotheses as well and hence not lie within the hypothesis space.

¹²Except for the dimensions of the search space introduced by the reordering strategy, which are not affected by the recombination strategy.

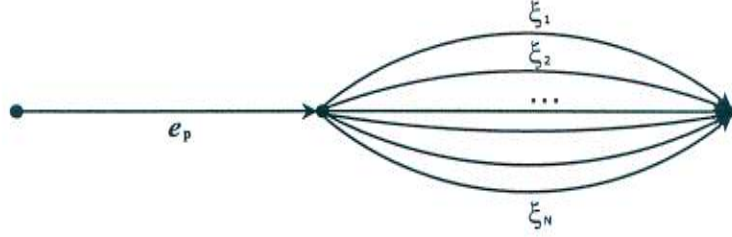


Figure 2.8: The lattice L_p shown here is trivial to traverse while still emitting the same hypothesis translations as the original lattice L , assuming that the partial hypothesis h_p (emitting the partial translation \mathbf{e}_p) holds.

Differently speaking, soundness means the optimum in inequality 2.5 is actually being reached, namely by expanding the given partial hypothesis over the edges as indicated by the backpointer information of the optimal final hypothesis found by the search algorithm.

$$\exists h \in \mathcal{H}(h_p, L) : \text{MED}(h) = \hat{d} \quad (2.6)$$

In order to prove both properties, we will firstly construct a translation lattice which is trivial to expand but provides the same (remaining) search space, for a given partial hypothesis, that is, emitting the same set of potential hypothesis translations.

For a given partial hypothesis h_p , let $\mathbf{e}_p = e_1^0 \dots e_{l_p}^0$ be the corresponding partial translation. A full expansion of the remaining search space $\mathcal{H}(h_p, L)$ leads to a finite set of translation hypotheses $\mathbf{e}_1 \dots \mathbf{e}_N$ all sharing the same prefix \mathbf{e}_p . Let $\xi_1 \dots \xi_N$ the corresponding set of suffixes:

$$\mathbf{e}_i = \underbrace{e_1^0 \dots e_{l_p}^0}_{\mathbf{e}_p} \underbrace{e_{l_p+1}^i \dots e_{l_i}^i}_{\xi_i} \quad \forall i = 1 \dots N \quad (2.7)$$

Then, the lattice L_p depicted in fig. 2.8 emits only and all elements of $\{\mathbf{e}_i : i = 1 \dots N\}$ (evidently, each path through the lattice emits $\mathbf{e}_p \cdot \xi_i = \mathbf{e}_i$ for some i , and for each i there is a path through L_p emitting $\mathbf{e}_i = \mathbf{e}_p \cdot \xi_i$).

The main difference between L_p and the original lattice under the assumption of the partial hypothesis h_p is that L_p emits the suffixes ξ_i which complete the translation hypothesis “at once”, while we need several sub-expansion and recombination steps in L . However, if we can proof soundness and completeness for the single-edge expansion lattice L_p , the proof easily carries over to the original lattice L under assumption of h_p :

PROOF Induction by the number of remaining edge (sub-)expansions in L . ■

We can therefore reduce the proof of completeness and soundness to the expansion of a single edge.

PROOF (*completeness*) Let \mathbf{m}_{best} be the last known row of the DP matrix as indicated by the partial, *recombined* hypothesis h_p (eq. 2.4). Let $\xi = e_{l_p+1} \dots e_l$ be any of the suffices associated to the edges in the lattice L_p (fig. 2.8). We can now complete the recombined DP matrix $\hat{M} = (\hat{m}_{ij})$ and calculate the total cost \hat{d} for the translation hypothesis $\mathbf{e} = e_1^0 \dots e_{l_p}^0 e_{l_p+1} \dots e_l$ against the fixed reference sentence.

Instead of a recombined hypothesis having a recombined DP matrix row, consider now the corresponding unrecombined hypotheses. Let h be such a partial hypothesis leading to a smaller Levenshtein distance $d < \hat{d}$ if expanded over ξ . Let $M = (m_{ij})$ be the corresponding DP matrix and let j be the column in which the Levenshtein path ϕ through M leaves row number l_p . Due to eq. 2.4, all entries of row number l_p of M are bigger than or equal to the entries in the corresponding row \mathbf{m}_{best} of \hat{M} . Hence, $\hat{m}_{l_p j} \leq m_{l_p j}$ and therefore, if we followed ϕ through \hat{M} , this would lead to a edit distance $d^* \leq d$.¹³ However, the Levenshtein path through \hat{M} is optimal in the sense that $\hat{d} \leq d^*$ and therefore $\hat{d} \leq d$, which contradicts our initial assumption about h .

As this holds for any suffix edge in L_p , eq. 2.5 holds for any path through L under the assumption of h_p (also see proof by induction over path length on p. 26), including the one found by the search algorithm. ■

PROOF (*soundness*) Let \hat{M} be the DP matrix as stated above with corresponding Levenshtein path $\hat{\phi}$ that leaves row l_p in column \hat{j} . Let h be the (unrecombined) partial hypothesis that contributed $\hat{m}_{l_p \hat{j}}$ in eq. 2.4. We have to show that h (without recombination) would have led to the same value \hat{d} .

Let $M = (m_{ij})$ be the corresponding DP matrix if we had not performed the recombination step. Assume this would lead (by performing the DP algorithm) to a Levenshtein distance d . As $m_{l_p \hat{j}} = \hat{m}_{l_p \hat{j}}$ by assumption, if we followed $\hat{\phi}$ through M , this would lead to the same Levenshtein distance \hat{d} . Hence, $d \leq \hat{d}$.

Now assume there would be another path ϕ^* through M leading to a Levenshtein distance $d^* < \hat{d}$. If ϕ^* leaves row number l_p in column j^* , ϕ^* would lead to an even smaller edit distance in \hat{M} as $\hat{m}_{l_p j^*} \leq m_{l_p j^*}$ by eq. 2.4. So ϕ^* would outperform the Levenshtein path in \hat{M} , which is impossible. ■

2.1.2 The PMED forced aligner

If we choose Position-independent Minimal Edit Distance as objective function, the resulting forced aligner differs in various ways from the Levenshtein forced

¹³Note that both matrices have the same hypothesis resp. reference words associated to their rows resp. columns, hence any (partial) path leads to the same additional costs (which are additively accumulated to the cost of the cell where the path starts) in both matrices.

aligner.

First of all, the Position-independent Minimal Edit Distance is much more simple to calculate: Due to the position-independence, all we need to know is the *count* of each word in the (target) vocabulary for both translation hypothesis and reference:

Definition 2.1 (Word counts)

Let $\mathbf{t} = (t_1, \dots, t_l)$ be a translation hypothesis of length l and let $\mathbf{r} = (r_1, \dots, r_m)$ be a corresponding reference sentence of length m . Furthermore, let \mathcal{V} be a vocabulary (set of words) containing all words occurring in \mathbf{t} respectively \mathbf{r} , then

$$N_{\text{Hyp}} : \begin{cases} \mathcal{V} & \rightarrow \mathbb{N}_0^+ \\ w & \mapsto \sum_{i=1}^l \delta_{w=t_i} \end{cases} \quad (2.8)$$

is the hypothesis word count of word w (number of times w occurs in the translation hypothesis), and

$$N_{\text{Ref}} : \begin{cases} \mathcal{V} & \rightarrow \mathbb{N}_0^+ \\ w & \mapsto \sum_{i=1}^m \delta_{w=r_i} \end{cases} \quad (2.9)$$

is the reference word count of word w (number of times w occurs in the reference sentence).

An error occurs if word counts for hypothesis and reference mismatch. If there are c more occurrences of word w in the reference sentence than in the translation hypothesis ($N_{\text{Ref}}(w) - N_{\text{Hyp}}(w) = c > 0$), this indicates c insertion errors.

$$e_{\text{ins}} := \sum_{\substack{w \in \mathcal{V} \\ N_{\text{Ref}}(w) > N_{\text{Hyp}}(w)}} N_{\text{Ref}}(w) - N_{\text{Hyp}}(w) \quad (2.10)$$

On the other hand, if the reference word count of a word w is smaller than the corresponding hypothesis word count by c occurrences ($N_{\text{Ref}}(w) - N_{\text{Hyp}}(w) = -c < 0$), this indicates c deletion errors.

$$e_{\text{del}} := \sum_{\substack{w \in \mathcal{V} \\ N_{\text{Ref}}(w) < N_{\text{Hyp}}(w)}} N_{\text{Hyp}}(w) - N_{\text{Ref}}(w) \quad (2.11)$$

As any pair of insertion and deletion error can be considered as a single substitution error (due to the position independence), the Position-independent Minimal Edit Distance can be calculated as

$$\text{PMED}(\mathbf{t}, \mathbf{r}) := \underbrace{e_{\text{ins}} + e_{\text{del}}}_{\text{counts substitutions twice}} - \underbrace{\min(e_{\text{ins}}, e_{\text{del}})}_{\text{substitutions}} \quad (2.12)$$

By distinguishing cases, this simplifies to

$$\text{PMED}(\mathbf{t}, \mathbf{r}) = \max(e_{\text{ins}}, e_{\text{del}}) \quad (2.13)$$

The structure of a partial (potentially incomplete) hypothesis is depicted in fig. 2.9. While developing the translation hypothesis, the decoder must keep

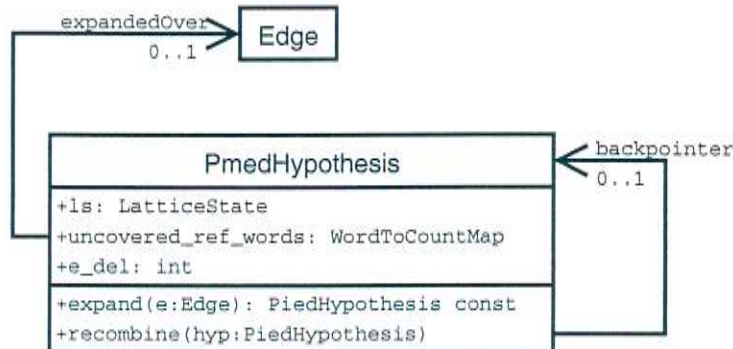


Figure 2.9: UML diagram of the translation hypothesis class of the PMED forced aligner.

track of the target words that haven't been covered yet (`uncovered_ref_words`). This way, when a partial translation hypothesis is expanded over a target word, the decoder can determine whether the word can be matched with a yet uncovered reference word of same type. Otherwise, a deletion error occurs, which are counted by `e_del` directly.

For a final hypothesis, the number of insertion errors e_{ins} (eq. 2.10) is the total number of uncovered reference words (sum over `uncovered_ref_words`). e_{del} can be read directly from `e_del`.

Note that unlike the hypothesis class of the Levenshtein forced aligner (fig. 2.5), `PmedHypothesis` has a single, unique backpointer for both the last hypothesis and the edge over which the last hypothesis had been expanded (except for the *initial hypothesis*, which has no backpointers). This makes the recombination step somewhat simpler, on the other hand, however, a massive, multidimensional recombination of the search space is not possible. Effectively, *pruning* becomes necessary during search, which introduces a trade-off between runtime and optimality of the solution.

2.2 Decoder Analysis: Taking the Decoder Apart

The main purpose of the forced aligner is to uncouple the lattice generation step from the lattice search step within the decoder (see sec. 1.4). The main questions a forced aligner can provide answers to are

- Search evaluation: *How good does the search algorithm perform?*

Oracle experiments can help us measure the loss of performance due to a suboptimal search, roughly denoted by the difference between the oracle score and the score of the path found by the ordinary decoder. Suboptimality of the search is mainly caused by *model errors*, that is, the hypothesis which is optimal with respect to the models is not truly optimal (with respect to the reference). Particularly, we have a model error

if the reference translation is considered less probable than the (wrong) decoded hypothesis.

The other possible error would be the search algorithm failing to maximize the model probability (search error), as indicated by a reference outperforming the decoded hypothesis (with respect to the models). The latter is usually caused by the reference not lying in the search space (e. g. monotone decoding although reordering is needed) or by the correct answer being pruned away (due to too restrictive beam sizes etc.).

If the forced aligner’s objective function matches the evaluation function, the oracle score is an upper bound for the decoder’s performance:

$$e(p_e(L)) > e(p_{\mathcal{M}}(L)) \quad \forall \mathcal{M} \quad (2.14)$$

where L is an translation lattice, e is an evaluation function positively correlated¹⁴ with translation quality, p_e is the oracle decoder optimizing for e and $p_{\mathcal{M}}$ is an ordinary decoder using a statistical model \mathcal{M} for the search.

- Lattice evaluation: *How “good” is our translation lattice?*
Using a forced aligner, we approximate an optimal model-free search. Hence, we can use the oracle score to compare the performance of various lattice generation techniques directly.
- Optimization: *How can we improve the translation quality of our system?*
Using the oracle score as a figure of merit for translation lattices, we can now search for methods to automatically improve the “lattice quality”. This should eventually result in better translation results.

Oracle scores like GWER are too optimistic. The last two points need some extra attention: One has to keep in mind that oracle decoders tend to systematically overestimate the decoder’s end-to-end performance. Naturally, they can give us an “upper bound” of what an optimal search could achieve¹⁵, but they usually fail to model the decoder’s behavior well (also see sec. 3.2.4).

As far as the optimization problem is concerned, oracle scores can easily be “tricked out”, in the sense that we can improve the oracle score of a lattice without improving (or even worsening) the decoder’s end-to-end performance. A trivial way to do this is to arbitrarily add edges. Evidently, this can only improve the quality of the oracle hypothesis (at least with respect to the objective function, eventually also with respect to the evaluation function if both are highly correlated), as the old optimal paths are still contained in the lattice. However, adding an edge increases the complexity of the search space, generally worsening the performance of the search and therefore potentially leading to a lower end-to-end performance of the decoder. Figure 2.10 depicts the dilemma. An extreme example would be to generate any possible target word (or any possible target phrase up to a reasonable phrase size) for each source word (resp. phrase) during lattice generation. This would result in an extremely dense lattice most likely to contain the full reference sentence (hence providing a perfect

¹⁴the higher, the better

¹⁵At least with respect to the objective function. The PMED forced aligner, for example, fails to achieve high BLEU performance (see sec. 2.4.2).

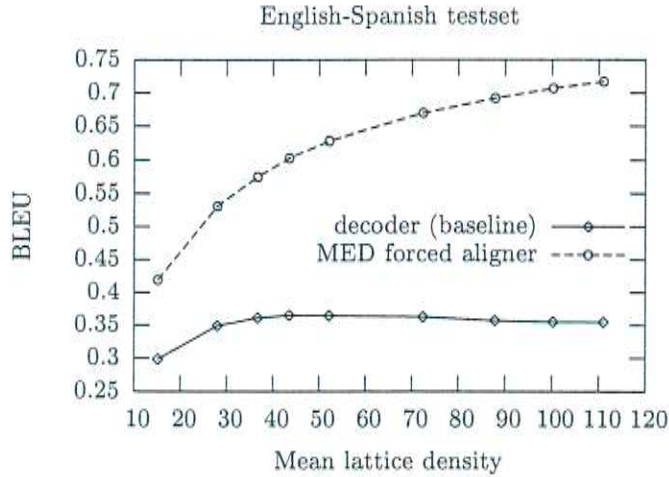


Figure 2.10: With increasing lattice density, the oracle score (using the Levenshtein decoder) monotonically increases, while the overall decoder performance slightly declines after a local optimum has been passed. A parameter limiting the number of extracted phrases during lattice generation, common to both forced aligner as well as decoder, was used to vary lattice density. Monotone decoding was used in either case.

translation) but also providing an extremely complex search space to the search algorithm. Differently speaking, if we do not take the *lattice complexity* into account, improving the oracle score does not necessarily imply an improvement of the end-to-end score.

A common figure of lattice complexity is the *lattice density*, that is, the (arithmetic) mean number of outgoing edges per node:¹⁶

$$\text{dens}(\underbrace{(\mathcal{N}, \mathcal{E}, n_0, \mathcal{F})}_{\mathcal{L}}) := \frac{|\mathcal{E}|}{|\mathcal{N}|} \quad (2.15)$$

Considering decoder performance with respect to lattice density leads us to *Performance-Density Graphs* (see sec. 2.3).

The task we face for the optimization problem is to increase the oracle score without increasing the lattice complexity too much. One way to do this is to create a dense lattice in the first place and then to apply pruning techniques in order to reduce the lattice complexity without losing too much oracle score performance.

An alternative approach of lattice evaluation and optimization that takes lattice complexity directly into account is introduced in chapter 3.

¹⁶An alternative approach to define lattice density would be $\sqrt[N]{\mathfrak{P}(L)}$ where $N = |\mathcal{N}|$ is the number of nodes in the lattice.

To be meaningful, oracle scores must at least be considered together with the corresponding lattice complexity

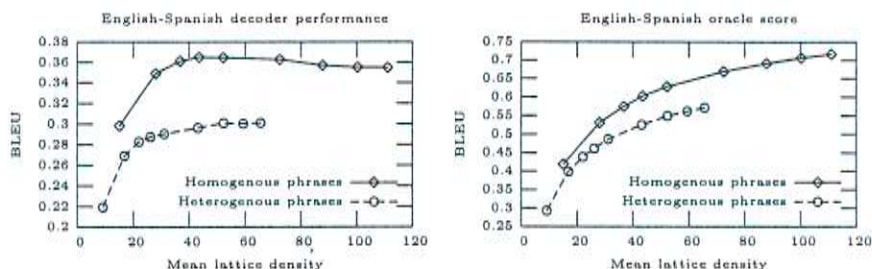


Figure 2.11: (Left) English-Spanish end-to-end performance of two decoders on an excerpt of the *European Parliament Plenary Sessions* (EPPS) corpus. One system was trained on the data the testset was extracted from and clearly outperforms an alternate system that used a corpus based on UN plenary session transcriptions for phrase extraction. (Right) Corresponding Levenshtein decoder performance. Compared to the decoder, the general shape of the curves is different (decaying vs. monotonically increasing). However, the main property of the EPPS system curve superseding the UN system curve all the way through could be preserved.

2.3 Performance-Density Graphs

A common way to compare systems independently from lattice complexity is to consider translation quality *with respect to* varying lattice density. If the curve of one system tends to lie below/above the curve of another system for a score positively correlated with translation quality, there is indication that the first system generally performs worse/better than the second one. Figure 2.11 gives an example for the scenario: As expected, the system trained on the data the test set was extracted from outperforms a system trained on in-domain data from a different source. The same observation can already be done on the lattice level, using the performance of the Levenshtein forced aligner.

Although performance graphs over lattice density meet our requirements in many ways, they still lack various features. First of all, they provide a rather pictorial figure which cannot easily be processed by machines e. g. for automatic optimization issues. Preferably, we would like a single real number (as this would automatically imply a total ordering on the score space). Possible ways to do this include:

- Normalize to a default lattice density.

In order to meet the default lattice density, we can either tune a corresponding parameter that varies the lattice density until the resulting lattice(s) have the expected density or interpolate the results from other densities.

This works well if we know that the behavior for a specific lattice density is representative for the whole lattice-density graph, i. e. curves do not intersect each other but lie either completely below or above each other. Relying on a specific density, however, attracts side-effects and negatively impacts robustness of the figure.

- Match the curve against a parametric class of curves

This approach is a well-known modeling technique, where we assume the curve / function in question to be of some certain class, e. g. Gaussians. The curve is then reduced to one or few parameters (e. g. mean and variance for Gaussians) which serve as a basis for a figure of merit for the curve.

Such *parametric learning* methods are known to generalize quite well, are hence robust against statistical noise and a comparison between the original curve and the best matching class member usually also provides a way to introduce confidence measures.

A tricky question, however, is the choice of the function class: While the curves produced by the forced aligner in figure 2.11 seem to belong to a more general function class of log-like functions, such an assumption would obviously lead to unsatisfying results for the corresponding curves produced by the ordinary decoder.

- Integration

By considering the area below the curve, we avoid side effects as potentially introduced by a dedicated lattice density, but also avoid model assumptions as in the “parametric learning” approach. An open questions here would be the optimal choice of integration boundaries.

All three approaches are based on performane-density graphs and thus share the property of needing a series of scores for varying lattice densities. However, any parameter in the decoder’s lattice generation module used for varying lattice density can cause side-effects when being modified. Furthermore, a series of scores require multiple evaluation and decoding runs, indicating excessive space and runtime requirements. A “more direct” approach (see sec. 3.3) could help to overcome these drawbacks.

2.4 Experiments

2.4.1 Optimality with respect to the objective function

We did some experiments in order to validate whether both forced aligners fulfill their specification and return a path which is optimal with respect to the particular objective function. Therefore we compared our baseline system with the output of the Levenshtein decoder using Minimal Edit Distance (fig. 2.12) and with the output of the PMED forced aligner using Position-independent Minimal Edit Distance (fig. 2.13). Both forced aligners clearly outperform their decoder pendants, which is what was wanted.

2.4.2 Optimality with respect to the evaluation metric

In the end, both forced aligners are supposed to return a path through the translation lattice which is optimal with respect to the evaluation function. If objective function and evaluation metric differ, this is not necessarily the case.

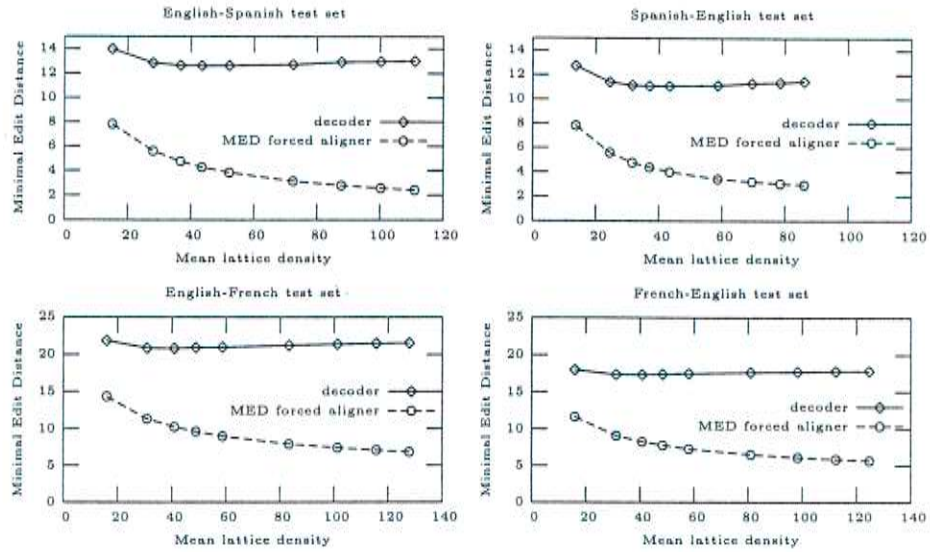


Figure 2.12: The hypotheses returned by the Levenshtein forced aligner are optimal with respect to Minimal Edit Distance. The Minimal Edit Distance of the hypothesis returned by the Levenshtein decoder denotes the Graph Error Rate (see p. 18) of the lattice.

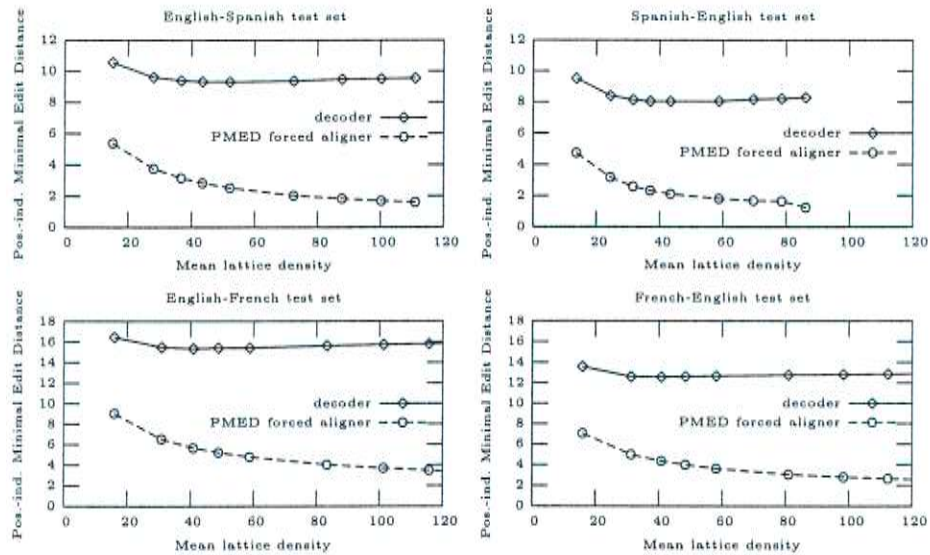


Figure 2.13: Although the PMED forced aligner cannot perform a full search, its oracle score is approximately a lower bound for the Position-independent Minimal Edit Distance an optimal path search could achieve.

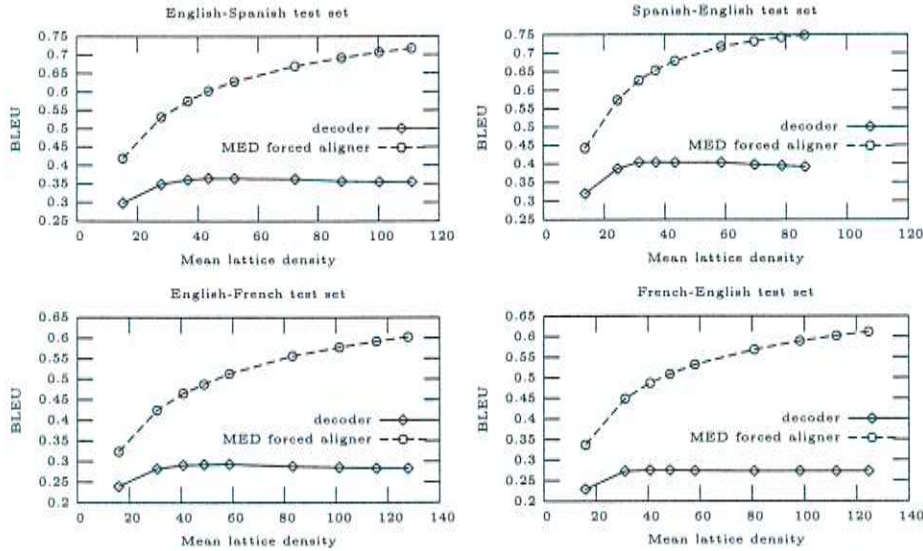


Figure 2.14: The Levenshtein forced aligner approximates an optimal search with respect to the evaluation function.

Compared to our ordinary decoder, the Levenshtein forced aligner clearly outperforms our baseline system (see fig. 2.14). We can assume the Levenshtein decoder at least approximates an optimal search with respect to the BLEU metric.

The hypotheses returned by the PMED forced aligner, however, fail to achieve high BLEU scores, or even outperform our baseline system (see fig. 2.15). Even worse, the oracle decoder’s (BLEU) performance starts to *decline* after a local maximum,¹⁷ which is rather untypical for oracle decoders. This is due to the fact that Position-independent Minimal Edit Distance and BLEU are less correlated than Levenshtein distance and BLEU are: Naturally, optimizing for PMED leads to a high unigram precision, as PMED forces the hypothesis to contain the same words as the reference does. However, n -gram precision can become arbitrarily small for $n > 1$, as PMED does not take word context into account at all. MED, on the other hand, prefers monotone alignments (see p. 19) which automatically leads to longer n -gram matches.

Table 2.1 shows an example of how n -grams contributed to the total BLEU score with respect to their length. As we can clearly see, the PMED forced aligner achieves a much higher unigram precision than the ordinary decoder (even higher than the Levenshtein decoder). However, the performance quickly drops for larger n -grams, which indicates that hypotheses returned by the PMED forced aligner tend to be “bags of words” that happen to contain the same words as the reference does, but lack a lot of monotonicity with respect to the reference.

¹⁷In this point, the PMED forced aligner’s oracle score shares a property with the decoder’s end-to-end performance, which could also be considered something positive. Also see remarks in section 3.3.7.

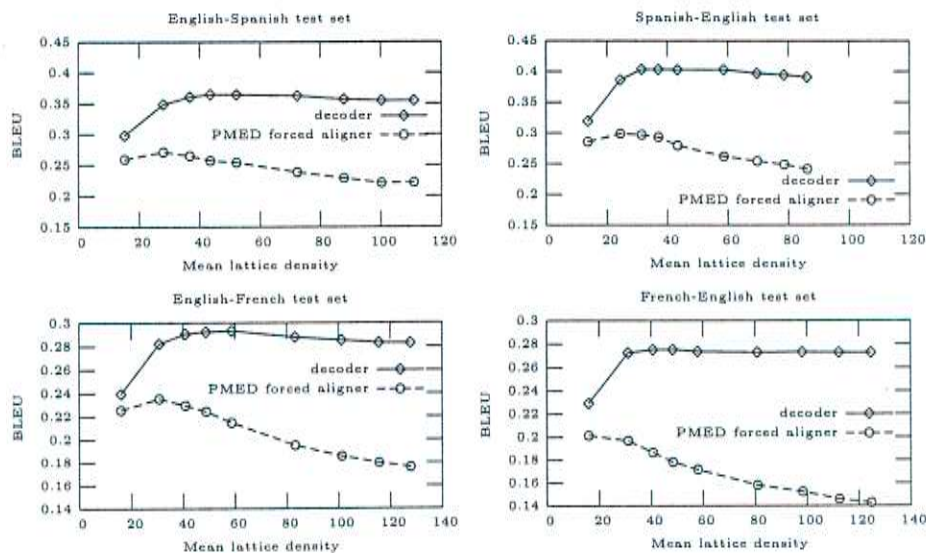


Figure 2.15: Evidently, optimizing for Position-independent Minimal Edit Distance does not imply optimality for BLEU. Also note that the PMED forced aligner’s performance declines with increasing lattice density, which is untypical for oracle decoders.

BLEU score by n -gram				
system	n -gram size			
	1	2	3	4
decoder (baseline)	0.6659	0.4175	0.2861	0.1993
oracle (MED)	0.8818	0.7675	0.6791	0.6034
oracle (PMED)	0.9662	0.3185	0.1358	0.0660

Table 2.1: This table shows how unigrams, bigrams, etc. contributed to the total BLEU score of the particular system. The baseline system with the highest lattice density was compared with the corresponding oracle systems.

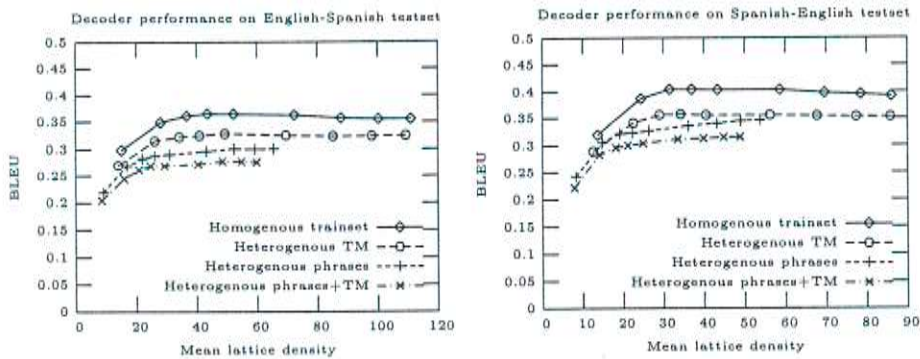


Figure 2.16: As expected, the decoder performs best if trained on a homogenous trainset.

The Levenshtein (MED) decoder, on the other hand, even scores significantly for longer n -grams.

2.4.3 Comparing systems using oracle scores

We have already seen that oracle scores can indeed help to distinguish “good” and “bad” systems (see fig. 2.11). We will refocus on this issue more systematically by modifying decoder parameters that alter the lattice structure and observe the behavior on both decoder and oracle performance.

For the English-Spanish task, we had two trainsets available (see tbl. 1.2), one based on the European Parliament Plenary Session (EPPS) and one based on the UN plenary sessions (UN). The English-Spanish test sets (one for each direction) are (disjoint) excerpts from the EPPS corpus so we can expect much better results from a system trained on the EPPS data than from a system trained on UN data.

We altered the baseline system in two ways: Firstly, we used the UN trainset for phrase extraction while the translation model (an IBM1 lexicon modeling word-to-word translation probabilities) was still trained on the EPPS data. A second alternate system also used the UN corpus in order to train the lexicon.

Recall that both forced aligners implement a model-free search. However, the translation model is already used during lattice generation (source-target phrase pairs are selected using their IBM1 forward and backward probabilities), so changing the translation model does change the lattice structure and therefore also the oracle decoder’s performance.

Figure 2.16 shows the performance of the English-Spanish baseline system for both directions. If we extract phrases from a heterogenous in-domain corpus (the UN corpus), we lose some performance, if we also train the word translation model on the UN corpus, we lose some more (which was what was expected).

Figure 2.17 shows the corresponding BLEU scores if we use the Levenshtein decoder instead of a statistical path search. Note that the generated lattices

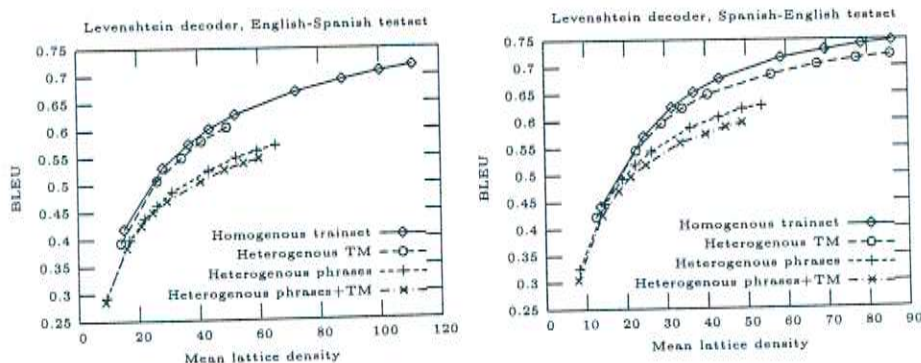


Figure 2.17: Ranking of the systems as indicated by fig. 2.16 can already be observed on the lattice level using the Levenshtein decoder's oracle score.

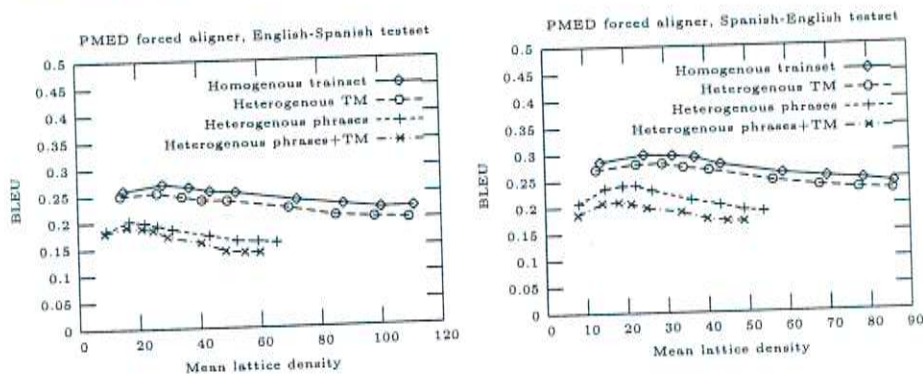


Figure 2.18: Although optimality with respect to PMED does not imply optimality with respect to BLEU, the three systems from figure 2.16 can yet be ranked the same way using the PMED forced aligner's oracle score.

are the same for both evaluation tracks. Using the Levenshtein decoder's oracle score leads to the same ranking of the systems as the one implied by the baseline system.

As we can see in figure 2.18, the ranking implied by the PMED forced aligner is also coherent with the observations from the baseline system. Although oracle hypotheses returned by the PMED forced aligner fail to achieve high BLEU scores, changes in the lattice structure that have a positive/negative impact on the decoder performance generally also have a positive/negative impact on the PMED forced aligner's oracle score (which is, as always, considered with respect to the lattice density).

Note that the difference between the baseline system and the alternate systems in figure 2.16 is just the change in the phrase extraction corpus and/or word translation model. Search parameters (esp. interpolation factors for the various models) had *not* been recalibrated. Now it is possible, of course, to recalibrate

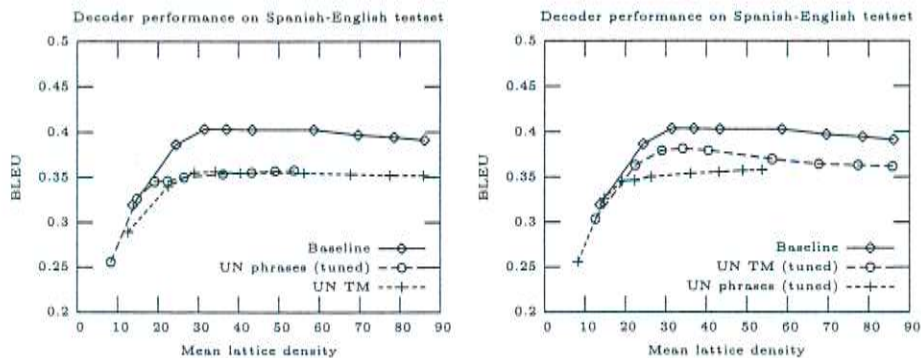


Figure 2.19: **(Left)** By lessening the influence of the language model, we can improve the UN-phrases system enough to outperform the UN-TM system. **(Right)** If we apply the changes of the search parameters on the UN-TM system as well, the UN-TM system outperforms the UN-phrases system again, in accordance to what was indicated by both oracle decoders.

the system where the phrases are extracted from the (heterogenous) UN corpus, which would result in the (tuned) UN phrases system outperforming the (un-tuned) UN translation model system (see fig. 2.19 **(Left)** for an example on the Spanish-English test set). The path search parameters, however, do not change the lattice structure, so the oracle scores of both forced aligners (which implement a model-free search) still would suggest that we lose more performance on using a heterogenous phrase extraction corpus than on training the word translation model on a heterogenous corpus. Such a comparison is, of course, illegal. If you compare decoder scores against oracle scores with varying lattices, you have to keep the path search parameters constant. Figure 2.19 **(Right)** shows that if the UN-phrases and UN-TM systems are compared using the same path search parameters, the UN-TM system still outperforms the tuned UN-phrases system, in accordance with the information provided by the oracle decoders.

2.5 Conclusions

We found that oracle decoders can help to assess MT systems independently from the path search.

Two oracle decoders for two different objective functions (*Levenshtein distance* and *Position-independent Minimal Edit Distance (PMED)*) were implemented.

The Levenshtein decoder performs a full search of the search space (except for the dimensions introduced by the reordering strategy), that is, the Levenshtein decoder implements a perfect, model-free search with respect to Minimal Edit Distance. Hypotheses returned by the Levenshtein decoder are approximately optimal with respect to the BLEU score, which causes the Levenshtein decoder to systematically overestimate the decoder's end-to-end (BLEU) performance. This effect becomes more evident with increasing lattice density. Differently

speaking, the Levenshtein decoder's oracle score generally features dense lattices to sparse ones (although dense lattices are harder to decode).

In general, oracle scores always have to be considered with respect to the corresponding lattice density.

The PMED forced aligner cannot perform a complete search of the search space, therefore introducing possible side effects due to the nonoptimality of the solution. Furthermore, hypotheses returned by the PMED forced aligner fail to be optimal with respect to BLEU, although having a high unigram precision. Actually, translations returned by the PMED decoder tend not to be useful at all, they are rather a bag-of-words containing the words of the reference sentence in arbitrary order. On the other hand, as opposed to the Levenshtein decoder, the PMED forced aligner does not systematically overestimate the decoder performance (at least with respect to the BLEU score), and it is still possible to use the oracle score in order to observe improvements and aggravations of the lattice structure. Furthermore, by objecting to the Position-independent Minimal Edit Distance, the decoder avoids some problems due to nonmonotone alignments.

Chapter 3

Advanced Lattice Evaluation

Many contemporary SMT decoders rely on *translation lattices* (see sec. 1.4). *Lattice evaluation* refers to making a reasonable statement about the “quality” of a lattice. A widely known approach to assess translation lattice quality has already been introduced in section 2.2. However, oracle scores like the Graph Error Rate (see p. 18) lack certain desirable features (see sec. 3.2.4), which makes them less useful for further applications, e. g. lattice optimization.

In the following sections, we will firstly outline some general ideas about translation lattice evaluation. After having a critical view on the Graph Error Rate resp. oracle scores in general, we introduce a novel, more direct metric for evaluating translation lattices called *Standard Word Count Distance (SWCD)*.

3.1 Translation Lattice Quality

To get an idea about the notion *quality* in this context, we consider some simple examples:

- A lattice containing only perfect paths (or even just one perfect path).
Obviously, this would be a perfect lattice by all means. From the path search’s point of view, this lattice is trivial to process and would always result in a perfect translation. On the other hand, the whole complexity of the translation problem is now left to the lattice generation: It is extremely difficult to construct such a lattice.
Quality by topology: The lattice contains only “good” paths
- A lattice containing all possible paths (i. e. translations).
In the first place, searching a “full” lattice seems to be a pointless task. There are only few good paths that have to be found out of a huge amount of paths that contribute nothing but statistical noise. However, if the models discriminate the set of good translation paths well against the vast majority of bad paths, an efficient search algorithm can still produce desirable results. Hence, most of the work is left to the statistical models and the search algorithm here.
Quality by geometry: The “good” paths (and only them) are short resp. have high scores

The size of the search space makes pruning techniques necessary that can (and at least occasionally will) prevent the search algorithm from finding the optimal path (with respect to the models). This usually leads to a trade-off between runtime and overall decoder performance.

On the other hand, if the statistical models do not help to distinguish the good peas from the bad ones, the lattice contains no or even misleading information about the true translation of the source sentence. In this case any search algorithm must fail to reliably find a good translation of the source sentence, and any lattice evaluation method should identify such a lattice as a “bad” lattice.

The complexity of the path search problem can be the basis for a figure of lattice quality, however, it is important to remark that “search problem complexity” must refer to the hardness of finding an optimal path *with respect to a reference* here as opposed to optimality with respect to model scores: In the example of the full lattice above, a model assigning probability 0 or 1 to arbitrary (!) edges leads to a trivial search problem¹ (just prune away the zero edges and take any path through the remaining lattice) but only the “good” model, assigning probability 1 to the “good” edges alone, produces lattices that result in a good overall decoder performance (which is what we really want)².

Quality by information

Another way to approach the topic is motivated by information theory: How much information does the lattice contain about the correct (i. e. reference) translation of the source sentence? How much *misleading* information does the lattice contain? Does the lattice encode the information in a very efficient or rather redundant way? How hard is it to reveal the information contained in the lattice? All these questions ask about properties of translation lattices that reflect different aspects of the notion “quality”. The lattice must contain all the information one needs in order to produce a correct translation, however, if some of the information is misleading, this might result in incorrect translations. Some redundancy might protect us from errors (caused by an imperfect path search and/or statistical noise), on the other hand, too much redundancy usually means waste of resources. Finally, the hardness of revealing the information contained in the lattice denotes the complexity of the path search problem stated before.

More precisely, structural properties of lattices indicating a high quality are

Precision The lattice should only produce events which are “good” translations of some source event (source words resp. phrases). Differently speaking, the target words emitted by a lattice should also occur in a valid translation (reference) of the source sentence. From the information theoretic point of view, this property corresponds to lattices not containing misleading information (see above). It is supposed to make sure that translations are *correct*.

Recall The lattice should produce enough events to cover a correct translation of the source sentence, including possible alternatives. High recall states that the lattice contains all the information one needs to produce

¹We did not take language models into consideration here. Alternatively, we could just pick one path and assign zero probability to all edges not lying on that path.

²Also see “model error” on p. 29

the translation. This property indicates whether or not translations are *complete*.

Compactness The lattice should not produce unnecessary events resp. the lattice should contain the necessary information to produce the translation, *but not more*. This property is known as “Occam’s³ Razor”⁴ which states that “non sunt multiplicanda entia praeter necessitatem”⁵. This principle seeks a balance between *precision* and *recall*.

Uniformity The lattice should cover possible reference translations uniformly. There is no use of making sure that a period is being translated into a period while an important low-frequency content word cannot be translated at all. This criterion is motivated by the fact that lattices usually cover reference events multiple times. It states that the lattice should not prefer one or the other word class (e. g. high frequency words against low frequency words).

If we also take the probabilities resp. costs associated to the lattice edges and nodes into account, we have to reinterpret these criteria in a probabilistic resp. cost-oriented context (e. g. how *probable* it is to produce only events that only occur in the reference, how *expensive* it is to cover all reference words, etc.).



William of Ockham,
 * ca. 1288 Ockham,
 England, † 9 April 1348
 Munich, Bavaria (now
 Germany). English
 Franciscan friar and philo-
 sopher. Summoned to
 Avignon in 1324 by Pope
 John XXII on accusation
 of heresy, he fled from
 Avignon in 1328 to seek
 the protection of Emperor
 Louis IV in Bavaria.
 [Wika]

3.2 Lattice Evaluation Metrics

Throughout this section, let \mathcal{L} be the set of possible translation lattices (def. 1.1). Furthermore, we will also consider lattices without weights:

Definition 3.1 (Lattices w/o edge weights)

For two lattices $L = (\mathcal{N}, \mathcal{E}, n_0, \Omega)$, $L' = (\mathcal{N}', \mathcal{E}', n'_0, \Omega')$ having the same lattice target vocabulary \mathcal{V} , define $L \sim L'$ iff

- there exists an isomorphism $\Phi : \mathcal{N} \rightarrow \mathcal{N}'$ with $\Phi(n_0) = n'_0$, and
- there exists an isomorphism $\Psi : \mathcal{E} \rightarrow \mathcal{E}'$ with

$$\Psi((n_i, n_j, e, \omega)) = (\Phi(n_i), \Phi(n_j), e, \omega')$$

(Two lattices are equivalent iff they have the same topology and corresponding edges emit the same target phrase, possibly at a different score.)

It is easy to see that \sim is an equivalence relation. We identify the system of all equivalence classes $\tilde{\mathcal{L}} = \mathcal{L}/\sim$ with the set of all “raw” lattices (without scores resp. weights associated to the edges).

³The correct spelling of this little town in Surrey, England, is “Ockham”.

⁴Although named after William of Ockham (~1288–1348), W. never stated the principle (which is much older than him) himself and used it rather implicitly in his scriptures. [Wikb]

⁵“plurality should not be posited without necessity” or — in everyday language — “the simplest solution is the best” [Wika]

3.2.1 Standard measures

Before we start exploring the scientific field of lattice evaluation, we will focus on various measures and concepts that reflect the criteria stated in section 3.1.

Precision and recall

Precision and *recall* as stated on page 42 are instances of a more general well-known concept:

Definition 3.2 (Precision and recall)

For a given set of test events and a corresponding set of reference events, we define

$$\text{prec}(t|r) = \frac{|r \cap t|}{|t|} \quad (3.1)$$

to be the precision of t (against reference r) and

$$\text{recall}(t|r) = \frac{|r \cap t|}{|r|} \quad (3.2)$$

to be the recall of t (against r).

In the case of Natural Language Processing (NLP), the set of test events t is usually the set of words or sub- n -grams contained in the test sentence (which is the translation hypothesis in case of MT). r is defined correspondingly.

There are various ways of how to define $|r \cap t|$. For NLP, the most common approach is to calculate the maximal duplicate-free unigram match between test and reference sentence, that is, the sum of minimal occurrences in t respectively r over all words occurring in both t and r :

$$|t \cap r| = \sum_{w \in \mathcal{V}_t \cap \mathcal{V}_r} \min(\#_t(w), \#_r(w)) \quad (3.3)$$

Precision denotes the relative amount of test events that actually “hit the target”. Recall, on the other hand, denotes the relative amount of reference events that have been covered by the test events. Both figures vary between 0 and 1 and describe antagonistic aspects of the similarity between t and r . It is usually easy to maximize one if you neglect the other: If we produce a lot of test events, for example, we can expect to have covered most of the reference events (which means a high recall), however, most of the test events will probably not occur in the reference (which implies a low precision). On the other hand, if we only produce events of which we are sure that they occur in the reference, the precision will most likely be close to 1, but we cannot expect a high recall in this case. The task is to maximize both measures simultaneously.

With respect to translation lattice evaluation, we will investigate how well translation lattice evaluation metrics model aspects of *precision* and *recall*.

Compactness

Compactness (see p. 43) in the context of translation lattices refers to the information content with respect to the lattice size; information content can already be expressed in terms of *precision* and *recall*, which leaves us to the task of metering the *size* of a lattice. Various measures provide a way to denote the lattice size, among them the number of nodes, the number of edges or the mean fan-out (see *lattice density*, p. 31) as a figure of *lattice complexity*.

The following criterion makes sure that for a metric m , the particular score of a lattice gives us a clue about the complexity or size the lattice could maximally have:

Definition 3.3 (Bounding lattice complexity)

Let $m : \mathcal{L} \rightarrow \mathbb{R}$ be a lattice evaluation metric positively correlated with lattice quality, and let $c : \mathcal{L} \rightarrow \mathbb{R}$ be a figure of lattice complexity (or size). m bounds lattice complexity (by c) iff for any $u \in \mathbb{R}$

$$\sup \{ c(L) \mid L \in \mathcal{L}, m(L) > u \} \quad (3.4)$$

exists. If m is a cost function (negatively correlated with lattice quality), we have to use $m(L) < u$ instead of $m(L) > u$ in eq. 3.4.

Uniformity

The *uniformity* criterion (see p. 43) is peculiar to translation lattices: We have to allow some redundancy in the lattices, that is, we must allow different lattice events to match one and the same reference event more than once (usually, a reference event can be matched by a test event only once when computing $|r \cap t|$). If we allow multiple matches, however, we postulate that all reference events should be treated equally. There exist standard measures for uniformity, like the *entropy* of a distribution:

$$\text{Ent}(\mathbf{p}) = \sum_{i=1}^m p_i \cdot \log p_i \quad (3.5)$$

where $\mathbf{p} = (p_1, \dots, p_m)$ is the distribution of coverage probabilities⁶ of the reference. $\text{Ent}(\mathbf{p})$ becomes maximal if, and only if, \mathbf{p} is the uniform distribution $p_i = 1/m \quad \forall i = 1 \dots m$.

3.2.2 Search algorithm independence

We expect a good lattice evaluation metric to measure the *impact of the lattice on the overall decoder performance*. This should be done independently from the actual search algorithm:

⁶ p_i denotes the probability that reference event i will be covered by a lattice event. A simple (model-free) way to estimate p_i is the number of lattice events that cover the reference event divided by the total number of lattice events.

Definition 3.4 A lattice evaluation metric $m : \mathcal{L} \rightarrow \mathbb{R}$ predicts decoder performance iff for any lattices $L, L' \in \mathcal{L}$

$$m(L) > m(L') \Rightarrow e(\hat{p}(L)) > e(\hat{p}(L')) \quad (3.6)$$

for a given best-path search algorithm $\hat{p}(\cdot)$ and translation evaluation method e (e. g. BLEU). We will call such a metric search algorithm independent (s.a.i.) iff (3.6) holds for any fixed search algorithm \hat{p} .

Although definition 3.4 does not state anything about the optimality of $\hat{p}(L)$ with respect to the models, we might want to restrict \hat{p} to “useful” search algorithms that at least approximately optimize a path through L with respect to the model scores. If the search algorithm is determined by a (multidimensional) parameter vector $\theta \in \Theta$, we write \hat{p}_θ and call a metric s.a.i. iff (3.6) holds for any $\theta \in \Theta$.

One should note that a lattice metric may depend on the translation evaluation metric e in equation 3.6, although independence is highly preferable. Anyway, the latter should follow from the fact that all translation evaluation metrics should be highly correlated.

model-independent lattice
evaluation metrics

An important design criterion for lattice evaluation metrics is whether to consider the model scores associated to the edges or not; in the latter case, we shall call such a translation lattice evaluation *model-independent*, as the metric ignores the statistical model(s) used by the path search.

If a metric m is model-independent, it is easy to see that $m : \tilde{\mathcal{L}} \rightarrow \mathbb{R}$, $\tilde{L} \mapsto m(L)$ for $L \in \tilde{L}$ is well-defined.

If we naïvely apply the s.a.i. criterion as stated in def. 3.4, model-independent evaluation metrics can never be s.a.i.: For any pair of lattices L, L' for which eq. 3.6 holds, we can easily construct (by inverting scores) lattices $\Gamma \sim L$, $\Gamma' \sim L'$ having the same topology (therefore still $m(\Gamma) = m(L) > m(L') = m(\Gamma')$) but $e(\hat{p}(\Gamma)) < e(\hat{p}(\Gamma'))$. We accomplished this by constructing artificial, degenerated models that gave paths in Γ a high score that had low scores in L and vice versa. Even worse, by choosing the statistical models carefully, we can make the search algorithm pick almost *any* path through the lattice.

In real life, of course, we only consider statistical models that are somehow similar by means of assigning high scores to “good” edges (respectively paths) and low scores to “bad” ones. Anyway, we can restate the criteria given in def. 3.4 for model-independent metrics by making the model scores part of the search algorithm (that is, the search algorithm takes the raw lattice, scores it, and tries to find the optimal path with respect to the scores). If we assume that the search algorithm at least fulfills the task of finding a path through the lattice maximizing the model score, we have

Definition 3.5 (s.a.i. for model-independent metrics)

A model-independent lattice evaluation metric $m : \tilde{\mathcal{L}} \rightarrow \mathbb{R}$ predicts decoder performance iff for any “raw” lattices $L, L' \in \tilde{\mathcal{L}}$

$$m(L) > m(L') \Rightarrow e(\hat{p}_{\mathcal{M}}(L)) > e(\hat{p}_{\mathcal{M}}(L')) \quad (3.7)$$

where $\hat{p}_{\mathcal{M}}(\cdot)$ returns the best path through a lattice with respect to the statistical model \mathcal{M} and e is a translation evaluation method (e. g. BLEU). We will call such a metric search algorithm independent (s.a.i.) iff (3.7) holds for any fixed statistical model \mathcal{M} .

Again, we might want to restrict \mathcal{M} to “useful” models. If \mathcal{M} is parametric with parameter θ , we write \mathcal{M}_{θ} and call m s.a.i. iff eq. 3.7 holds for any $\theta \in \Theta$.

The quality of the statistical models significantly impacts the overall decoder performance and should therefore influence a lattice quality metric, following the statements above. Therefore, it is questionable whether it makes sense to consider model-independent metrics at all. By actually performing a search and evaluation step, we get a (model-dependent) lattice evaluation metric that trivially predicts decoder performance:

$$m_{\mathcal{M}}(L) := e(\hat{p}_{\mathcal{M}}(L)) \quad (3.8)$$

So why bother with model-independent metrics at all? The appealing property of model-independent lattice evaluation metrics is that only they decouple the decoder’s lattice generation step from scoring step and the search for an optimal path (with respect to the model scores). This makes it possible to optimize lattice generation (i. e. pruning methods) and path search (i. e. model weights) independently.

We can trivially evaluate a translation lattice model-dependently by performing a full decoding run.

3.2.3 Model consistency

The s.a.i. criterion states that the lattice metric should increase (respectively decrease) with increasing (decreasing) overall decoder performance if varying over different lattices. If we sort the lattices by density (respectively generate lattices with varying density) and plot both end-to-end decoder performance as well as lattice metric in a performance-density graph (see 2.3), the graphical interpretation of the s.a.i. criterion is that both decoder performance curve as well as lattice metric curve should share the same *shape* (i. e. they should increase and decrease together). Above all, they should have local optima at the same place.⁷

As this should hold for *any* model \mathcal{M} , a model-independent s.a.i. lattice evaluation metric requires some amount of *consistency* from the family of allowed models: If the performance-density graph (or any other plot of performance with respect to a lattice property) of the lattice evaluation metric in question should look similar to the decoder PDG for any (useful) model, all the PDGs caused by the various models have to look similar themselves. More precisely,

Definition 3.6 (Consistency)

A family of statistical models \mathfrak{M} is consistent (with respect to the overall decoder performance) iff for any “raw” lattices $L, L' \in \tilde{\mathcal{L}}$ and $\mathcal{M}, \mathcal{M}' \in \mathfrak{M}$

$$e(\hat{p}_{\mathcal{M}}(L)) > e(\hat{p}_{\mathcal{M}}(L')) \Rightarrow e(\hat{p}_{\mathcal{M}'}(L)) > e(\hat{p}_{\mathcal{M}'}(L')) \quad (3.9)$$

⁷This criterion is not restricted to performance-density graphs. Similarity of shape should hold for plots over *any* lattice property.

Consistency of \mathfrak{M} is a necessary and sufficient criterion for the existence of a model-independent s.a.i. translation lattice evaluation metric:

PROOF *Consistence is necessary:* Let m be a model-independent s.a.i. lattice evaluation metric. Furthermore, assume \mathfrak{M} to be not consistent. Hence, there exist $L, L' \in \tilde{\mathcal{L}}$, $\mathcal{M}, \mathcal{M}' \in \mathfrak{M}$ with

$$e(\hat{p}_{\mathcal{M}}(L)) > e(\hat{p}_{\mathcal{M}}(L')) \quad \wedge \quad (3.10)$$

$$e(\hat{p}_{\mathcal{M}'}(L)) \leq e(\hat{p}_{\mathcal{M}'}(L')) \quad (3.11)$$

Without loss of generality, assume $m(L) > m(L')$ (otherwise exchange \mathcal{M} and \mathcal{M}'), then eq. 3.11 contradicts the s.a.i. criterion from eq. 3.7.

Consistence is sufficient: For any fixed model $\mathcal{M}_0 \in \mathfrak{M}$, \mathfrak{M} consistent, $e(\hat{p}_{\mathcal{M}_0}(\cdot))$ as in eq. 3.8 is a model-independent (!) s.a.i. lattice evaluation metric. The s.a.i. property follows directly from the consistency criterion. The metric is model-independent as the fixed model \mathcal{M}_0 is independent from the actual model the decoder uses for the path search. ■

Roughly speaking, model consistence states that some part of the overall decoder performance depends on the “raw” lattice topology, independently from the choice of the statistical model (that is, improvements in lattice topology do hardly impact improvements in model quality and vice versa). In that case, the s.a.i. criterion for model-independent metrics states that the metric focuses on that model-independent “core” of the decoder performance.

Furthermore, the proof of sufficiency indicates that in case of consistency, any model \mathcal{M}_0 can act as a representant for the whole consistent family of models \mathfrak{M} , leading to a family of *canonical lattice evaluation metrics*:

$$\mathfrak{C} := \{\hat{p}_{\mathcal{M}}, : \mathcal{M} \in \mathfrak{M}\} \quad (3.12)$$

That means, if the models are consistent, we can just pick any (suboptimal) model independently from what the optimal model would be and optimize the lattice generation technique (respectively pruning parameters) with respect to overall decoder performance (using the fixed model). On the other hand, if the models are not consistent, there exists no model-independent s.a.i. lattice evaluation metric at all!

Experiment results as shown in fig. 3.1 look promising: variations of the statistical model change the major properties of the performance-density graph of the corresponding translation system only slightly.

We will close this section by showing that if a family \mathfrak{M} of models is consistent, then a model-independent lattice evaluation metric m is already s.a.i. if m predicts decoder performance (eq. 3.7) for *any* $\mathcal{M} \in \mathfrak{M}$:

PROOF Let \mathcal{M}_0 be a model for which eq. 3.7 holds. Then, for any “raw” lattices $L, L' \in \tilde{\mathcal{L}}$ and any $\mathcal{M}' \in \mathfrak{M}$:

$$m(L) > m(L') \quad \Rightarrow \quad e(\hat{p}_{\mathcal{M}_0}(L)) > e(\hat{p}_{\mathcal{M}_0}(L')) \quad (3.7)$$

$$\Rightarrow \quad e(\hat{p}_{\mathcal{M}'}(L)) > e(\hat{p}_{\mathcal{M}'}(L')) \quad (3.9)$$

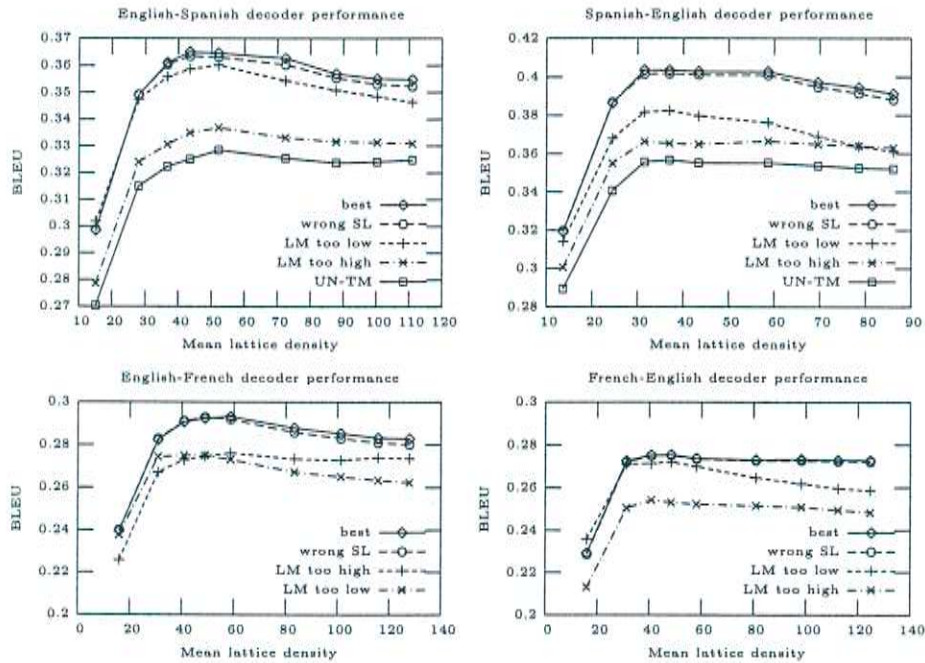


Figure 3.1: As expected, varying the model (while keeping the lattice topology unchanged) changes the decoder's performance-density graph, however, the similarity between the various curves for the four test sets is evident. Models were varied by either setting interpolation factors for the various submodels (language model, word translation model etc.) to different values or by using different submodels itself (e. g. an IBM1 lexicon trained on UN data for the EPPS testset).

Therefore, eq. 3.7 holds for *any* $\mathcal{M} \in \mathfrak{M}$ which is the s.a.i. criterion from def. 3.5. ■

3.2.4 Graph Error Rate and Graph Word Error Rate

The *Graph Word Error Rate (GWER)* (see p. 18) is a lattice oracle score well-known in NLP. As we have seen (p. 19), the Levenshtein decoder can be used to approximate the Graph Word Error Rate. Furthermore, the hypothesis returned by the Levenshtein decoder can be used to (exactly) compute a slightly different oracle score called *Graph Error Rate* (see p. 2).

Unfortunately, the behavior of the Levenshtein decoder differs in major points from the decoder with respect to performance (see sec. 2.2), which means the Levenshtein decoder resp. GER cannot predict decoder performance (and therefore never be s.a.i.). Another technical flaw of oracle scores in general is that only edges on the optimal path contribute to the score, however, a lattice with many good paths should have a much higher score than a lattice with one good path and a lot of noise otherwise. Differently speaking, adding noise to a lattice worsens its quality and should therefore impact any lattice evaluation score negatively⁸. On the other hand, adding “good” translations (that is, translations close to the reference translation(s)) improves the lattice quality (apart from the increased lattice complexity resp. density which is usually something negative) and should therefore positively impact the lattice score in general.

It would be wrong to say that the edges not lying on the optimal path did not influence the oracle score at all: They influence the optimal path (and therefore the lattice score) by *potentially* lying on the optimal path. However, this effect comes only into play for larger statistical contexts: If we modify the lattice generator to produce slightly less edges (which lowers the lattice density and therefore improves the lattice quality⁹), *some* lattices might lose edges on their optimal path leading to a slight drop in the overall oracle score (and therefore, on the other hand, worsening the lattice quality). For a *single* lattice, however, these effects are much harder to observe. By exploiting the information contained in a lattice more efficiently, we should be able to reduce the amount of needed testing data.

3.2.5 The canonical metric

For a consistent family of models, any canonical metric (eq. 3.12) is s.a.i. by definition.

Even if the models are not consistent, we can still get useful results by toggling between a lattice optimization step (adjusting the pruning parameters assuming fixed model interpolation factors) and a search optimization step (optimizing the model interpolation factors assuming a fixed lattice topology (i. e. fixed pruning parameters etc.), using well-know methods like minimum error rate training). Considering the lattices as “hidden variables”, this approach is known as the

⁸Note that in the model-dependent case, we can, without loss of quality, allow some noisy edges if they are clearly marked as noise by the model

⁹Recall that oracle scores have to be considered together with a measure for lattice complexity (see p. 30).

EM algorithm, which is a standard approach if part of the training parameters cannot be observed directly.

On the other hand, a full decoder run is comparatively slow. This is especially severe if we apply the EM algorithm, which requires several iterations in order to converge. Furthermore, canonical scores share some flaws with oracle scores: Just as any oracle score is directly influenced by the edges on the optimal path only, a canonical score is (directly) impacted by the edges on the *decoded* path only. Hence, in order to make a statement about *lattice quality*, the canonical score uses the information contained in a lattice inefficiently, just as oracle scores do.

3.2.6 Novel approaches

The various disadvantages of oracle and canonical lattice evaluation metrics motivated the search for a completely novel way of how to make a reasonable statement about “lattice quality”. First of all, note that many of the problems with oracle and the canonical scores stem from the fact that both approaches just pick one path out of the lattice (either by a full decoding run or an oracle experiment) and use an existing string-to-string similarity metric (like BLEU, MED or WER). Instead, we wish to directly take advantage from the full information provided by the whole lattice.

single path evaluation

A first, naïve approach motivated by the string-to-string(s) BLEU metric was called *Edge Precision*. Here, we considered each sub- n -gram (up to a maximal size \hat{n}) of the target phrase associated with each edge of the lattice. Each of these n -grams either occurred in the reference sentence (match/hit) or they didn’t (miss). The final score was a weighted sum of the various hit rates with respect to the n -gram length, as collected over a given test set.

This new approach had several advantages, compared to the single-path evaluation approaches offered by an (oracle) decoder:

- + All the edges of a lattice contribute to the overall score directly. Differently speaking, adding a noisy edge producing only target words that do not occur in the reference spoils the hit rate statistics and therefore lowers the overall score; adding perfect edges that produce phrases that occur in the reference sentence cause a series of hits for all contained n -grams and therefore improve the overall score; in the general case, some of the sub- n -grams of a lattice edge’s target phrase will occur in the reference sentence, and some (especially the larger ones) won’t, assessing the positive and negative aspects of a target phrase (with respect to the reference sentence) simultaneously.
- + The calculation of the metric is linear in time with respect to the number of edges in the lattice and therefore very fast. However, note that the runtime is approximately exponential with respect to the largest n -gram size \hat{n} .
- + By the way the metric was designed, we can make a statement about how a single edge impacts the total score: The more sub- n -grams of the target

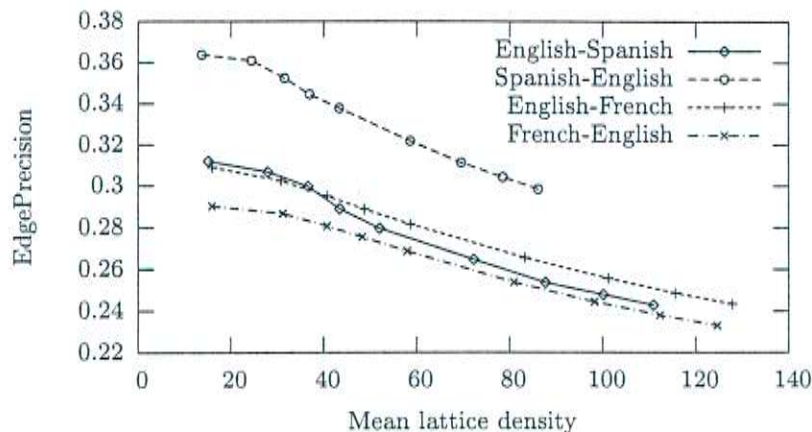


Figure 3.2: Performance (with respect to the “Edge Precision” metric) Density Graph for the various test sets. Maximal n -gram length was $\hat{n} = 4$, using uniform weights of $\lambda_i = 0.25, i = 1, \dots, 4$ for the weighted hit rate mean.

phrase associated to an edge match the reference sentence, the better the edge.

In practice however, the metric performed less well than expected. Fig. 3.2 shows a plot of the behavior of the Edge Precision metric with respect to varying lattice density for our four testsets (tbl. 1.3). Note that all the plots are more or less straight, monotonically decreasing lines. However, a straight line contains hardly any information (except for the slope). Compared to the performance-density graph of the end-to-end decoder performance, the characteristic drop in performance for small (sparse) lattices is missing. Neither the metric itself, nor any simple linear transformation is s.a.i.

We can try to modify the metric’s tuning parameters in order to improve the metric’s expressiveness. We already know that *data sparseness* is a common phenomenon in model training (see p. 1.3.5), especially for large n -grams. The effect of unseen events is even more severe in an evaluation scenario, as we try to find n -grams in a single¹⁰ sentence instead of a big (but never big enough) training corpus. We can therefore try to prefer shorter n -grams, for which more statistical evidence is available.

However, as fig. 3.3 shows, modifying the weight factors does not change much in principle. Actually, the fact that any configuration of $\vec{\lambda}$ results in a constant (with respect to lattice density) shift is quite surprising in the first place.

This effect becomes clear if one dissects the Edge Precision metric into its components by setting $\lambda_i = \delta_{i=j}, (j = 1, \dots, \hat{n})$. As figure 3.4 shows, the n -gram hit rates themselves hardly differ by more than a constant shift, so any interpolation of the four functions cannot result in more than a constant shift. One could argue that we might get a more satisfying metric if we combine the hit

¹⁰In many evaluation scenarios (including BLEU), multiple references are used for this reason.

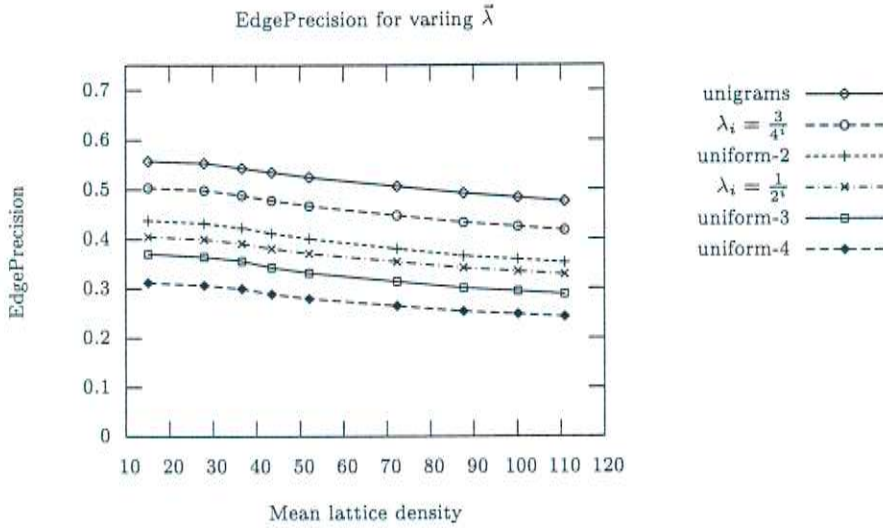


Figure 3.3: Edge Precision for the English-Spanish testset with varying hit rate interpolation factors. “uniform- n ” stands for $\lambda_i = \frac{1}{n}$ for $i \leq n$, $\lambda_i = 0$ otherwise.

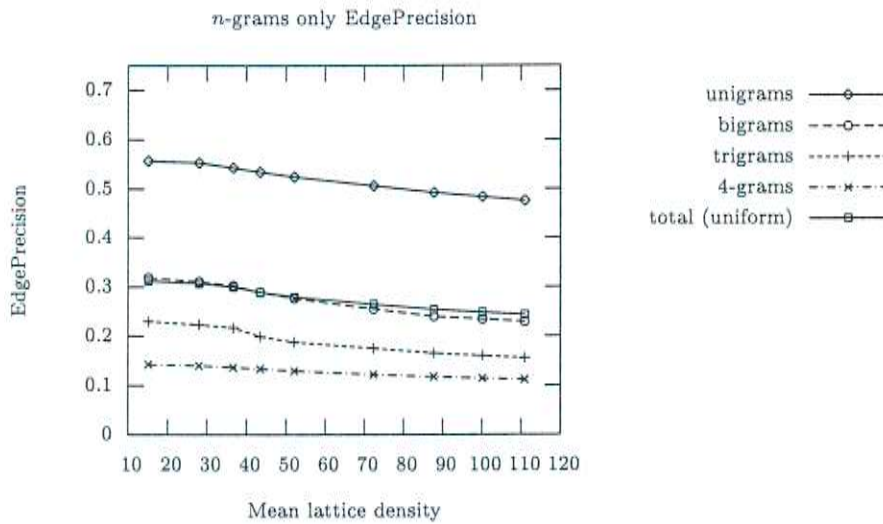


Figure 3.4: Components of the Edge Precision metric

rates in a *non-linear* way. However, there are mainly two reasons why this is not an option:

Firstly, there surely is a non-linear way to combine the various hit rates (or even use further information like the number of nodes etc.) and get a metric that nicely predicts the end-to-end decoder performance, at least on the development set. However, we might run the risk of “tuning” our metric to the desired behavior (*overfitting*). In this case, the metric’s behavior would be rather unpredictable on an independent test set, respectively in “real use”. Instead of just “picking” a function that meets our requirements, all design decisions must be justified by rational argumentation.

Secondly, the Edge Precision metric has some structural flaws that cannot be solved by simple means. Most obviously, the Edge Precision metric makes sure that the events (*n*-grams) produced by the lattice also occur in the reference. However, the Edge Precision does not guarantee that the reference events are produced by the lattice at all. Edge Precision is a pure *precision* metric, it does not take the lattice-reference pair’s *recall* (see p. 44) into account¹¹. For instance, a lattice producing only a period would most likely lead to a perfect unigram precision (if we assume that most reference sentences are ended by a period), although the lattice hardly contains any information about the correct translation at all.

All these issues led to a radical redesign of our lattice evaluation metric. Keeping the aspects of translation lattice quality as outlined in section 3.1 (p. 42) in mind, we setup the following design criteria (also see sec. 3.2.1):

The metric should

- be model-independent,
- at least roughly predict decoder performance
- penalize noisy edges (precision),
- penalize lattices that do not cover the reference(s) well, that is, completely (recall) and uniformly.

To account for *compactness* seemed to be a secondary objective to us, as non-compact lattices rather influence the decoder’s need for resources than the decoder’s end-to-end performance. Compared to the decoder’s end-to-end performance, we consider memory usage and runtime to be a minor problem at the moment, although there might be scenarios where these issues gain importance (like translation software on handheld devices etc.).

3.3 Standard Word Count Distance (SWCD)

In this section, we will introduce a model-independent lattice evaluation metric called *Standard Word Count Distance (SWCD)* and examine the metric with

¹¹BLEU, for instance, solves this problem by allowing a reference *n*-gram to be matched only once. Furthermore, a *brevity penalty* makes sure that hypotheses are long enough (see sec. 1.5). Therefore, hypotheses of sufficient length that do not cover the reference sentence well either include events not seen by the reference or include reference events more than once, both leading to a *n*-gram mismatch.

respect to various of the previously stated criteria. SWCD is a cost function negatively correlated with lattice quality (the lower, the better), therefore we will consider the negated score in order to predict decoder performance.

Definition 3.7 (Lattice target word counts)

Let $L = (\{n_0, \dots, n_k\}, \mathcal{E}, n_0, \{n_k\})$ be a translation lattice with lattice target vocabulary \mathcal{V}_L , and for any edge $E_j \in \mathcal{E}$, let $\mathbf{t}_j := (t_1^j, \dots, t_{l_j}^j)$ be the target phrase associated to the edge. Furthermore, let $\mathcal{V} \supset \mathcal{V}_L$ be a vocabulary containing all occurring (target) words of L . We extend definition 2.1 (hypothesis word counts, reference word counts) by lattice target word counts

$$N_{\text{Lat}} : \begin{cases} \mathcal{V} & \rightarrow \mathbb{N}_0^+ \\ w & \mapsto \sum_{E_j \in \mathcal{E}} \sum_{i=1}^{l_j} \delta_{w=t_i^j} \end{cases} \quad (3.13)$$

(total number of times w occurs on the target side of an edge).

The basic idea of the SWCD score is to meter the similarity between the reference and the lattice count distribution. One way to do this is to calculate the squared error between reference and lattice word count distribution:

$$\sigma^2 = \sum_{w \in \mathcal{V}} (N_{\text{Ref}}(w) - N_{\text{Lat}}(w))^2 \quad (3.14)$$

where \mathcal{V} contains both lattice and reference vocabulary.

However, one has to take care of the fact that the lattice word counts will generally be much higher than the reference counts. We can model this by using a scaled word count distribution λN_{Ref} instead of N_{Ref} and by optimizing λ with respect to a minimal squared error:

$$\hat{\lambda} \stackrel{!}{=} \arg \min_{\lambda} \underbrace{\sum_{w \in \mathcal{V}} (\lambda N_{\text{Ref}}(w) - N_{\text{Lat}}(w))^2}_{\sigma_{\lambda}^2} \quad (3.15)$$

As $\sigma_{\lambda}^2 \rightarrow \infty$ for $|\lambda| \rightarrow \infty$, we are looking for a local minimum:

$$\begin{aligned} \frac{\partial \sigma_{\lambda}^2}{\partial \lambda} &= 0 \\ \Leftrightarrow \sum_{w \in \mathcal{V}} 2 \cdot (\lambda N_{\text{Ref}}(w) - N_{\text{Lat}}(w)) \cdot N_{\text{Ref}}(w) &= 0 \end{aligned}$$

which can be simplified to a unique optimal solution for λ :

$$\hat{\lambda} = \frac{\sum_{w \in \mathcal{V}} N_{\text{Lat}}(w) \cdot N_{\text{Ref}}(w)}{\sum_{w \in \mathcal{V}} N_{\text{Ref}}^2(w)} \quad (3.16)$$

The reference must not be empty for eq. 3.16 to be well-defined.

SWCD needs references not to be empty

$\hat{\lambda}$ indicates how many times a reference word is covered on average. We will call $\hat{\lambda}$ the *lattice redundancy*.

Redundancy is usually a good thing, as it enables the path search to achieve its goal on multiple ways. However, $\hat{\sigma}^2$ becomes small if, and only if the lattice

covers all reference words (and only them) similarly well. In fact, it is easy to proof that $\hat{\sigma}^2$ becomes 0 if, and only if, the ratio between reference word counts and lattice word counts is constantly the same for all $w \in \mathcal{V}$ that occur in L and/or the reference. In this case, this constant is $\hat{\lambda}$:

PROOF

$$\begin{aligned}
\hat{\sigma}^2 = \sigma_{\hat{\lambda}}^2 &= 0 \\
\Leftrightarrow \sum_{w \in \mathcal{V}} \underbrace{(\hat{\lambda} N_{\text{Ref}}(w) - N_{\text{Lat}}(w))^2}_{\geq 0} &= 0 \\
\Leftrightarrow (\hat{\lambda} N_{\text{Ref}}(w) - N_{\text{Lat}}(w))^2 &= 0 \quad \forall w \in \mathcal{V} \\
\Leftrightarrow \hat{\lambda} N_{\text{Ref}}(w) - N_{\text{Lat}}(w) &= 0 \quad \forall w \in \mathcal{V} \quad (3.17) \\
\Leftrightarrow \frac{N_{\text{Lat}}(w)}{N_{\text{Ref}}(w)} &= \hat{\lambda} \quad \forall w \in \mathcal{V} \quad (3.18)
\end{aligned}$$

Note that, if $N_{\text{Ref}}(w)$ were 0 in eq. 3.18, $N_{\text{Lat}}(w)$ would be 0 as well (following from eq. 3.17). However, we consider only words in \mathcal{V} that occur in the lattice and/or the reference at least once. ■

Generally, we want to allow more redundant lattices to commit more errors. Furthermore, we must account for the fact that longer lattices (having more nodes as a result of more source words) naturally produce more errors than small (short) lattices. Thus, we finally can define the SWCD score to be the average standard error (that is, the square root of the squared error $\hat{\sigma}^2$) normalized by redundancy and the number of source nodes:

Definition 3.8 (SWCD score) Let $(L_i, \mathbf{r}_i)_{i=1}^S$ be a set of S translation lattices together with a reference sentence. Furthermore, for each lattice-reference sentence pair (L_i, \mathbf{r}_i) , let $\hat{\sigma}_i^2 = \sigma_{i, \hat{\lambda}_i}^2$ be the minimal squared error together with its corresponding redundancy $\hat{\lambda}_i$ as defined in eq. 3.15 respectively eq. 3.16. Let $|L_i|$ denote the number of nodes in translation lattice i . Then we define

$$\begin{aligned}
\text{SWCD}((L_i, \mathbf{r}_i)_{i=1}^S) &= \frac{1}{S \cdot \overline{|L|}} \cdot \sum_{i=1}^S \frac{\sqrt{\hat{\sigma}_i^2}}{\hat{\lambda}_i} \\
&= \frac{1}{\sum_{i=1}^S |L_i|} \cdot \sum_{i=1}^S \frac{\sqrt{\hat{\sigma}_i^2}}{\hat{\lambda}_i} \quad (3.19)
\end{aligned}$$

where

$$\overline{|L|} := \frac{1}{S} \cdot \sum_{i=1}^S |L_i| \quad (3.20)$$

is the mean number of nodes per lattice.

Note that we did not normalize each standard error by the number of nodes in the corresponding lattice individually in order to avoid putting too much weight on short lattices. Lattice-reference pairs resulting in a very low redundancy can also cause trouble, see section 3.3.2 for details.

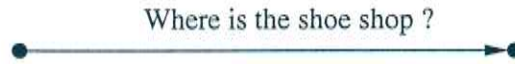


Figure 3.5: A trivial lattice for a single reference sentence.

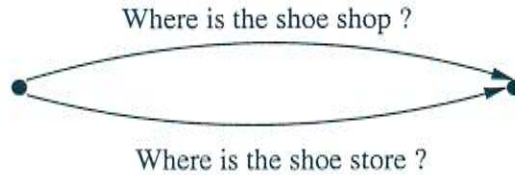


Figure 3.6: A trivial lattice for two reference sentences.

3.3.1 Multiple references

The previously introduced approach can easily be extended for multiple references. First note that a single reference sentence can be represented by a trivial lattice as shown in figure 3.5. We can then use lattice word counts (eq. 3.13) instead of reference word counts (eq. 2.9) and compare two lattices instead of a lattice and a sentence.

Having two references means that we have two alternate ways to translate a sentence. Figure 3.6 shows an example lattice that encodes this fact. However, this naïve approach leads to a very redundant lattice. Instead, we would expect a “perfect” translation lattice to encode the information common to all references in a more compact way (fig. 3.7).

This intuitively motivates a redefinition of reference counts as follows:

Definition 3.9 (Word counts for multiple references)

Let $\{\mathbf{r}_i : i = 1 \dots n\}$ be a set of reference sentences $\mathbf{r}_i = (r_{i1}, \dots, r_{im_i})$ and let \mathcal{V} be a vocabulary containing all occurring reference words. Then the reference word count for multiple references is defined to be

$$N_{\text{Ref}} : \begin{cases} \mathcal{V} & \rightarrow \mathbb{N}_0^+ \\ w & \mapsto \max_{i=1}^n \sum_{j=1}^{m_i} \delta_{w=r_{ij}} \end{cases} \quad (3.21)$$

Note the max operator in equation 3.21. This way we prefer compact lattices that do not produce the easy part of the translation over and over again. We also penalize lattices that correlate well with only one reference but fail to produce alternatives.

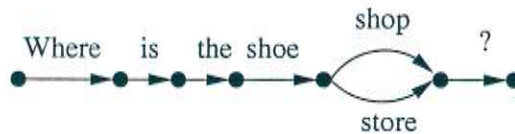


Figure 3.7: An optimal lattice for two reference sentences.

An alternate approach would be to sum up reference word counts, which can be emulated by concatenating the reference sentences to one long sentence.

3.3.2 Bounding the redundancy

Equation 3.19 fails if the redundancy $\hat{\lambda}$ becomes 0 for at least one lattice-reference pair. This is the case if, and only if, the lattice does not produce any reference words on its target side (lattice and reference vocabulary are disjoint):

PROOF

$$\begin{aligned} \hat{\lambda} &= 0 \quad (3.16) \\ \frac{\sum_{w \in \mathcal{V}} N_{\text{Lat}}(w) \cdot N_{\text{Ref}}(w)}{\underbrace{\sum_{w \in \mathcal{V}} N_{\text{Ref}}^2(w)}_{>0}} &= 0 \quad \text{ref. nonempty} \\ \sum_{w \in \mathcal{V}} \underbrace{N_{\text{Lat}}(w) \cdot N_{\text{Ref}}(w)}_{\geq 0} &= 0 \quad \Leftrightarrow \\ N_{\text{Lat}}(w) = 0 \quad \vee \quad N_{\text{Ref}}(w) = 0 \quad \forall w \in \mathcal{V} \end{aligned}$$

■

This phenomenon mainly occurs for small lattices (having small lattice vocabularies, caused by low density (sparse lattice) and/or few nodes (short source sentences)) and/or short references (having a small reference vocabulary).

But even if lattice and reference vocabulary are not disjoint, $\hat{\lambda}$ can become arbitrarily small, causing the quotient in eq. 3.19 to become arbitrarily large, hence a single lattice-sentence pair can falsify the whole test set result.

In order to solve this problem, we can specify a fixed lower bound $\underline{\lambda}$ for $\hat{\lambda}$

$$\hat{\lambda} := \min(\underline{\lambda}, \hat{\lambda}) \quad (3.22)$$

and use $\hat{\lambda}$ instead of $\hat{\lambda}$.

Figure 3.8 shows a plot of the SWCD performance of the English-Spanish test set over increasingly dense lattices. $\underline{\lambda}$ varies between 0 and 1. Note that the bound for $\hat{\lambda}$ only affects small, sparse lattices. In the whole test set, there are only two segments that produce a redundancy smaller than 1 (and are therefore effected by $\underline{\lambda}$). Although those two segments modify the total test set score only slightly, the curve becomes a little bit smoother with $\underline{\lambda} \nearrow 1$.

In the Spanish-English direction, we hardly observed any artefacts caused by low redundancies. So specifying a lower bound for redundancy $0 < \underline{\lambda} \leq 1$ virtually did not change anything, but it did not hurt either (see fig. 3.9).

For the English-French test set, we observed a very special behavior in both directions, as can be seen in fig. 3.10 and fig. 3.11. It turned out that the test set contained a misaligned sentence pair: The English sentence “(applause)” (3 words) was aligned to French sentence containing 103 words.

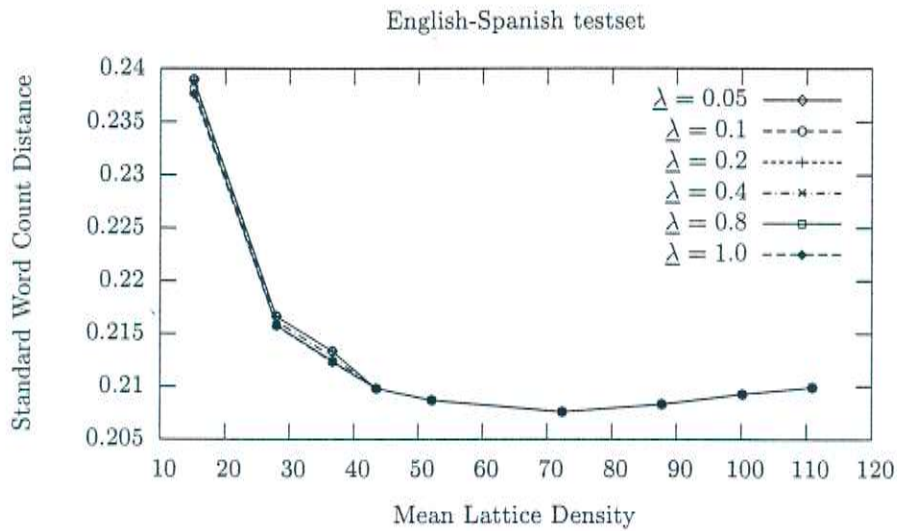


Figure 3.8: Different values for λ on the English-Spanish test set. The first three curves ($\lambda = 0.05 \dots 0.2$) lie on each other.

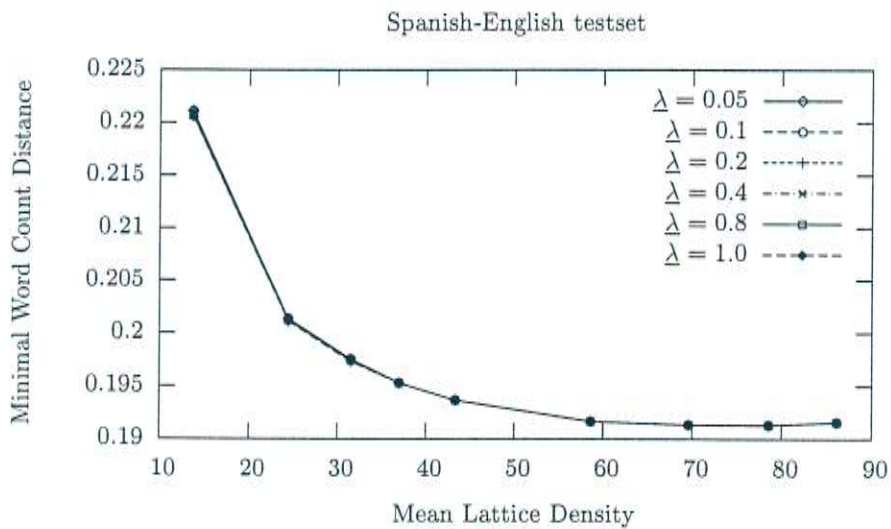


Figure 3.9: In the Spanish-English direction, introducing a lower bound for redundancy hardly changed anything. However, the system curve was smooth and neat in the first place anyway.

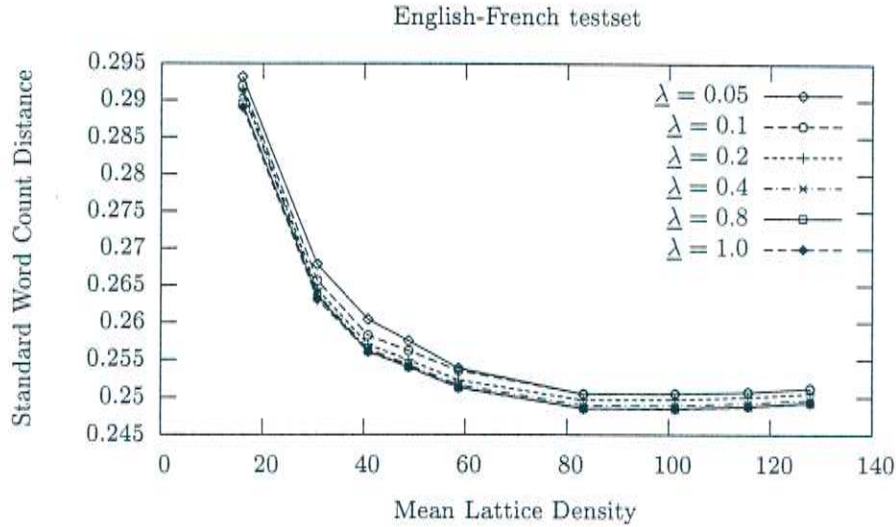


Figure 3.10: Varying λ over the English-French test set for the English-French direction. The test set contained a misaligned sentence pair.

For the English-French direction, the misaligned sentence falsified the test set result as follows: From the third data point in fig. 3.10 (at a mean lattice redundancy of ~ 40) onwards, the misaligned (small) English lattice and (long) French reference sentence was the only lattice-sentence pair that still produced a redundancy smaller than 1; all other lattices had managed to cover enough reference words to produce reasonable values for $\hat{\lambda}$ by then. Due to the length of the reference sentence (the redundancy is divided by the squared sum over the reference word counts, see eq. 3.16) and the few target words produced by the lattice, hardly matching any of the reference words¹², the redundancy of the particular lattice-reference pair remained very small throughout all experiments. At the fifth data point, the redundancy $\hat{\lambda}$ of the said lattice-reference pair managed to supersede 0.1, which caused the curve for $\lambda = 0.05$ and $\lambda = 0.1$ to collapse. After that, varying λ mainly varied the squared error $\hat{\sigma}^2$ of the misaligned lattice-sentence pair by $\lambda \cdot \sum N_{\text{Ref}}^2(w)$, which caused a shift in the order of $(\sqrt{\sum N_{\text{Ref}}^2(w)}) / (\sqrt{\lambda} \cdot \sum |L_i|)$ on the whole test set score (see eq. 3.19), independently from the particular lattice density.

In the French-English direction, the situation was different: The squared error sum was mainly contributed to from the side of the lattice, that is

$$\hat{\sigma}^2 \approx \sum N_{\text{Lat}}(w)$$

which grows with increasing lattice density but is mainly independent from λ . Also note that the lattice (and therefore also $\hat{\sigma}^2$) was quite large even for low

¹²The French sentence neither contained parentheses nor the French word for “applause”, however, with growing lattice density the lattice acquired edges producing periods and commas and other high-frequency words (due to wrong phrase alignments, e. g. [“applause” / “applaudissement, ”]), which — of course — occurred in the (long) French reference sentence.

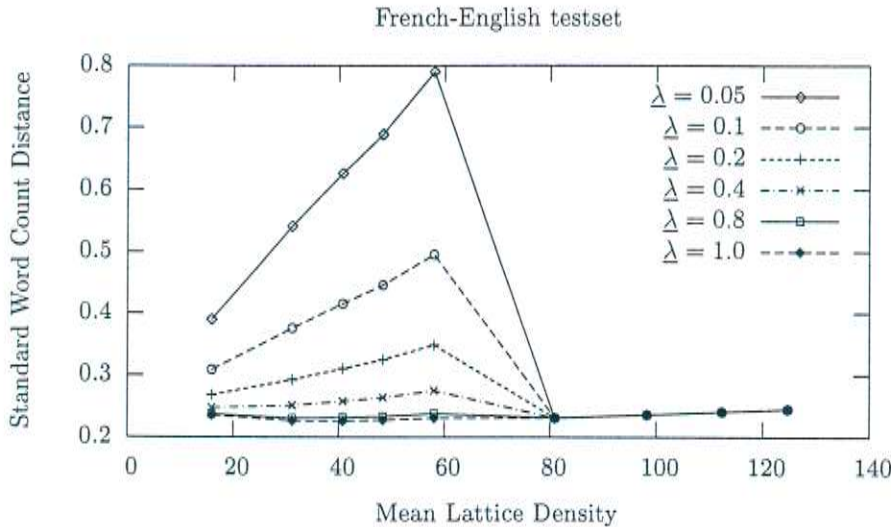


Figure 3.11: Varying λ over the English-French test set for the French-English direction. The effect of the misaligned sentence pair can be clearly observed on the whole test set.

densities, due to the high number source words (represented as nodes in the lattice). Furthermore, due to the much shorter reference sentence (having a squared word count sum of 3), the lattice-sentence pair had a significant chance of resulting in a reasonable value for $\hat{\lambda}$ (i. e. greater than one) as soon as the lattice would manage to cover some of the reference words by any chance (especially the high-frequency words "(" and ")"). Until then, however, adding edges would just monotonically increase the squared lattice count sum and therefore the squared error. Dividing by a very small λ (dividing by 0.05 is equal to multiplying by 20) intensified this effect so much that it could easily be observed on the whole test set score.

From what we have seen in our experiments, we propose a minimal redundancy of $\lambda = 1.0$. This strategy not only helps to reduce the inadequately high impact of lattice-reference pairs having a very low redundancy, it is also intuitively sound: We make sure that every unmatched reference word contributes to the squared error sum with at least one error.

In the case of the misaligned sentence in the English-French test set, we could not avoid a constant shift depending on λ in the French-English direction (or, in general, when a large source sentence is misaligned to a short reference sentence). However, for a large enough λ , the SWCD Performance-Density Graph falls back into its usual general shape (increasing on both ends) which is what we expect most. In the English-French direction, a minimal redundancy of 1.0 prevented the misaligned sentence pair from falsifying the whole test set score completely. Simply speaking, choosing a minimal redundancy helps to stabilize the SWCD score against the malicious impact of misaligned sentences in the test set.

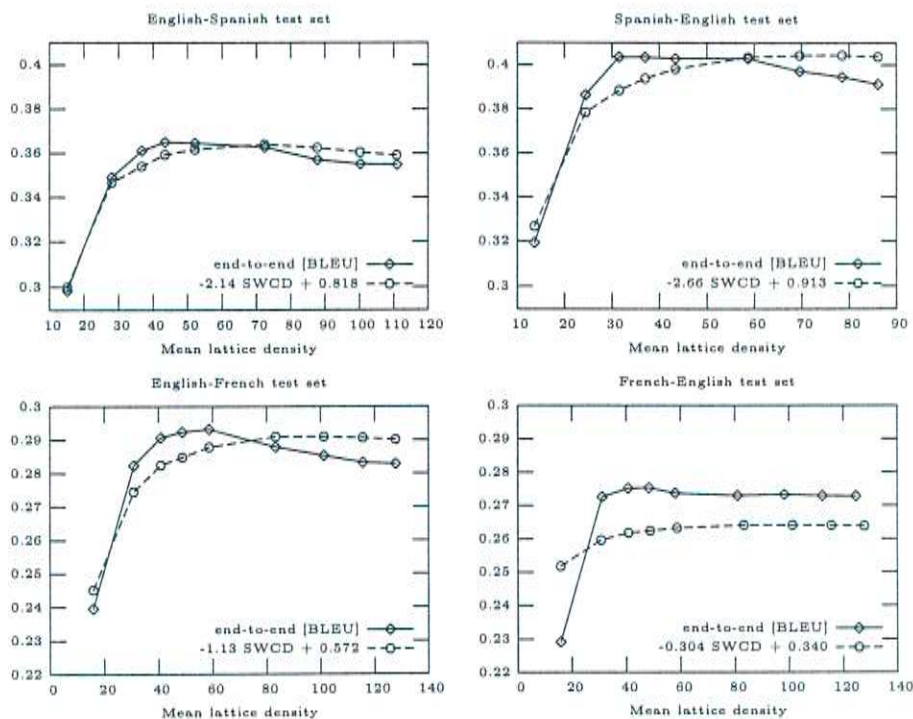


Figure 3.12: Comparing end-to-end decoder performance (BLEU) against a linear transformation of the Standard Word Count Distance. The coefficients of the linear transformation were computed for each test set individually using the best linear fit minimizing the squared error. As SWCD is a cost function (in contrast to BLEU), the stretching factor must be negative.

3.3.3 Predicting decoder performance

In order to check if our new metric predicts end-to-end decoder performance (def. 3.4, p. 46), we compare the Performance-Density Graph of the baseline system (using BLEU as performance metric) against the plot of the SWCD score computed on the same set of lattices. As SWCD does generally not operate on the same scale as the BLEU score, we compare the BLEU curve against a linear transformation $\Phi(x) = m \cdot x + c$ of the corresponding SWCD plot for each of the four test sets (see fig. 3.12). Note that linear transformations are generally bijective (except for the mappings on a constant), so no information is added or lost. The coefficients of the linear transformation were computed individually for each testset using the best linear fit minimizing the squared error. Note that BLEU is a figure of merit, while SWCD is a cost function, so the scaling factor m is always negative.

Fig. 3.12 shows that SWCD generally does a good job of correlating with the end-to-end performance of the decoder. There is, however, still room for improvement: Above all, SWCD tends to slightly overrate dense lattices (compared

to the decoder), which causes SWCD to reach its optimum at a later point with respect to increasing lattice density.

However, compared to oracle scores like Graph Error Rate (see p. 18) SWCD is definitely to be preferred (see fig. 2.10).

3.3.4 Precision

We expect our metric to penalize noisy edges (see p. 54). With respect to a reference, “noisy” means that an edge emits words respectively a phrase that has nothing to do with the reference sentence.

Note that due to eq. 3.16, only words that co-occur in both lattice and reference can increase the lattice redundancy. So adding noisy edges does not or hardly increase lattice redundancy (note that adding an edge can never *decrease* lattice redundancy), while the squared error *does* change: Assuming that in the best case, every added noisy word is a different one, the squared error sum grows by 1 for each added noisy word. Further assuming a constant mean number c of target words per edge, $\sqrt{\hat{\sigma}^2}$ grows at least to the order of $\sqrt{c \cdot n}$ in the number of added (noisy) edges. With $\hat{\lambda}$ being approximately constant, we conclude that SWCD’s growth is unbounded when adding noise to the lattice, which is what was wanted.

3.3.5 Recall and uniformity

Precision forces the lattice to produce events (that is, in our case, unigrams) that also occur in the reference. However, if we allow lattices to cover reference events multiple times, we have to make sure the lattice does not cover the same reference event over and over again (e. g. a lattice producing only periods or other high-frequency words).

Generally, lattice words cooccurring in the reference cause the redundancy to increase (the nominator increases in eq. 3.16) which leads to a decrease of the quotient $\sqrt{\hat{\sigma}^2}/\hat{\lambda}$. Furthermore, we have already seen that $\hat{\sigma}^2$ becomes 0 if, and only if, the reference words are covered uniformly (see p. 56).

On the other hand, if we keep covering a reference word w^* although the reference contains at least one different word w' , we have

$$\sqrt{\hat{\sigma}^2} > \sqrt{(N_{\text{Lat}}(w') - \hat{\lambda}N_{\text{Ref}}(w'))^2} \approx \hat{\lambda} \cdot N_{\text{Ref}}(w')$$

if $\hat{\lambda}$ grows big enough.¹³ Differently speaking, $\sqrt{\hat{\sigma}^2}$ grows linearly with $\hat{\lambda}$ for every poorly covered reference word, which causes SWCD to prefer lattices that cover the reference completely and uniformly.

3.3.6 Lattice complexity

With regard to the lattice’s compactness (see sec. 3.2.1), one should note that SWCD does *not* bound lattice complexity (neither by density nor by the number

¹³Note that w^* hardly contributes to the squared error due to the choice of $\hat{\lambda}$.

of edges; see def. 3.3, p. 45): For a given reference, any lattice containing an arbitrary number of n edges that go from start node to end node and just emit the reference sentence results in a redundancy $\hat{\lambda} = n$ and a squared error $\hat{\sigma}^2 = 0$, therefore having a perfect Standard Word Count Distance of 0. Hence, the supremum in eq. 3.4 does not exist for any $u > 0$.

3.3.7 Unigrams and position independence

fluency: measure for the hypothesis translation of well-formedness in the target language

Evidently, the lattice evaluation metric defined in eq. 3.19 only takes unigrams into account. However, putting words into context by considering their neighbors is necessary in order to assess the *fluency* of the hypothesis translation. Furthermore, multigrams can help to properly align the hypothesis sentence with the reference translation and therefore detect translation errors that produce a word that happens to occur somewhere else in the reference (but in another context).

Differently speaking, the SWCD metric bears the potential flaw that the information about the word position within the lattice is lost; we might want to match a target word at the beginning of the lattice (e. g. assigned to an edge coming out of the root node) to a reference word at the beginning of the reference sentence, but not to a word at the reference sentence's very end.

Most string-to-string similarity metrics use multigram precision (e. g. BLEU, p. 14) and/or word position (e. g. Levenshtein distance, p. 17) in order to compare two sentences (a counterexample would be the Position-independent Word Error Rate (PER), see p. 18).

We have to keep in mind, however, that the evaluation task of translation lattices is different by nature. First of all, we do not know yet at what potential position a target word will occur in the translation hypothesis. Actually, how far a word can travel heavily depends on the decoder's reordering strategy (p. 11). Thus, if we let the lattice metric depend on the target word's position in the translation lattice, we implicitly restrict the decoder to a certain reordering strategy which may or may not be desirable.

A similar argument holds for multigrams. Which words occur next to each other in the translation hypothesis depends both on the chosen path and the rearrangement of the phrases applied by the reordering strategy. The only multigram information intrinsically contained by a translation lattice is given by multiword phrases associated to a single edge, given that the decoder leaves their structure intact¹⁴. However, as we have seen with our "Edge Precision" approach (p. 51), a n -gram based approach does not necessarily lead to a useful metric. Furthermore, fig. 3.4 indicates that most occurrence information is already provided by unigram frequency.

Also recall that the PMED forced aligner provides useful results (see p. 38) even though the PMED score is position-independent and only considers unigrams as well (see sec. 2.5).

¹⁴We could, for example, assume a post-processing step where the decoder rearranges words in order to implement a global reordering strategy. This is, however, rather exotic and we could evaluate against a modified pre-postprocessed reference in this case as well.

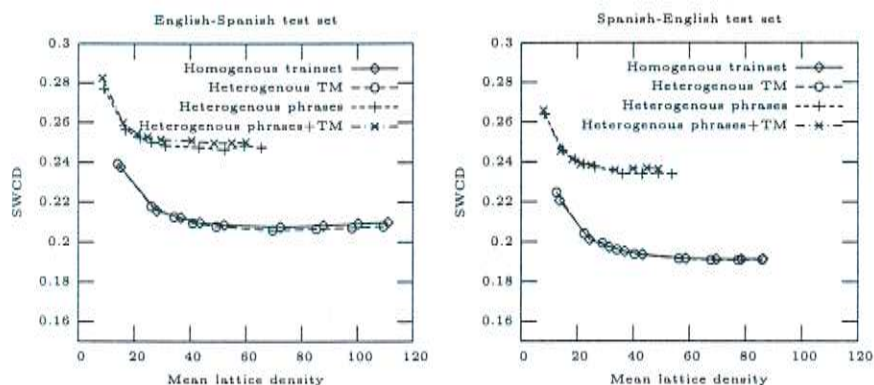


Figure 3.13: Corresponding SWCD performance of the systems assessed in section 3.3.8 (also see fig. 2.16).

3.3.8 Comparing systems using SWCD

We repeated the experiments we did for the two forced aligners (see sec. 3.3.8), now using SWCD scores. We can see various properties of the end-to-end decoder score reflected in the SWCD performance-density graph:

- + Using a heterogenous parallel corpus for phrase extraction impacts the system by far more negatively than using a heterogenous training corpus for the translation model.
- + For small lattices, performance improves quickly with growing lattice density. After having reached a local minimum resp. maximum, the performance flattens.
- + The Spanish-English system generally performs better than the English-Spanish system (with respect to BLEU).

There are, however, also some discrepancies worth mentioning:

- SWCD reaches its optimum slightly later than the end-to-end decoder performance. Furthermore, the optimum seems to be less developed (although, in many cases, the maximum of the end-to-end decoder performance hadn't been very well-developed either).
- The difference between the baseline system and the UN-TM system is much smaller than in the decoder PDG. In fact, on the Spanish-English test set, the difference is virtually nonexistent, for the English-Spanish test set, the UN-TM even slightly outperforms the baseline system. One way to explain this, is that in the baseline system, the modified translation model is used both for phrase selection and for the path search.¹⁵ On the other hand, the lattices themselves, and therefore also SWCD, are

¹⁵At the time of writing, the STTK decoder did not provide a way to specify different lexica for phrase selection and edge scoring when using online phrase extraction.

only influenced by the modified TM during the phrase selection phase. Hence, we could conclude that — due to a modified TM — it is during the path search that we lose most performance. However, this observation is inconsistent with the results we had with both oracle decoders (see sec. 3.3.8), where we clearly observed a drop in performance on the lattice level after switching to the modified TM.

3.4 SWCD Oracle Pruning

In this section, we will derive a oracle lattice pruning algorithm based on the lattice evaluation metric introduced in section 3.3. The main objective of such an oracle pruning method is to use the oracle decisions (of whether to prune a lattice edge or not) as training data for non-oracle pruning methods which will be applied during runtime.

In a first step, we will rank the edges of a lattice by the magnitude of improvement their removal would bring on the lattice-reference pair's score. In a second step, we will select all edges for pruning for which the improvement on the total lattice-edge pair's score lies over a certain threshold.

Either the hard decisions of the pruning criterion can be used for training (as positive and negative examples, as used in perceptron learning, backpropagation etc.) or, alternatively, we can use the hypothetical score improvement of the lattice edges directly (e. g. backpropagation again (predict the score improvements directly), minimum error rate training etc.).

3.4.1 How a single edge changes the metric

In this section, we will analyze how the removal of a single edge impacts the two main statistics used by the SWCD score, the redundancy λ and the minimal squared error $\hat{\sigma}^2$.

For any lattice $L = (\{n_0, \dots, n_k\}, \mathcal{E}, n_0, \{n_k\})$ with corresponding reference R , let \mathcal{V} be a vocabulary covering all reference and lattice target words and let N_L resp. N_R be the corresponding lattice and reference count function. Furthermore, let

$$\mathcal{V}_L := \{w \in \mathcal{V} : N_L(w) > 0\} \quad (3.23)$$

$$\mathcal{V}_R := \{w \in \mathcal{V} : N_R(w) > 0\} \quad (3.24)$$

be the lattice target vocabulary respectively reference vocabulary (now only containing words that really occur in the lattice resp. reference).

For any edge $E = (n_i, n_j, \mathbf{t}, \omega)$ in L , we can define an edge word count function

$$N_E : \begin{cases} \mathcal{V} & \rightarrow \mathbb{N}_0^+ \\ w & \mapsto \sum_{i=1}^l \delta_{w=t_i} \end{cases} \quad (3.25)$$

where $\mathbf{t} = t_1 \cdots t_l$, $t_i \in \mathcal{V}$ is the target phrase associated to the edge. As before, N_E induces an edge vocabulary

$$\mathcal{V}_E := \{w \in \mathcal{V} : N_E(w) > 0\}. \quad (3.26)$$

Note that $|\mathcal{V}_E| \ll |\mathcal{V}_L|$.

We assume that the unpruned lattice L has a redundancy $\hat{\lambda}$ and a minimal squared error of $\hat{\sigma}^2$. If we prune any edge E from the lattice, we get a new lattice L_E having $\hat{\lambda}'$ and $\hat{\sigma}^{2'}$ as new redundancy respectively new minimal squared error.

Instead of actually removing E from L , we can modify the lattice's counts directly as only the lattice's target word counts are used for calculation of the SWCD score. Note that

$$N_L(w) = \sum_{E \in \mathcal{E}} N_E(w) \quad \forall w \in \mathcal{V} \quad (3.27)$$

and therefore

$$N_{L_E}(w) = N_L(w) - N_E(w) \quad \forall w \in \mathcal{V} \quad (3.28)$$

for any edge E in L . Using the definition of redundancy (eq. 3.16) we get

$$\begin{aligned} \hat{\lambda}' &= \frac{\sum_{w \in \mathcal{V}} N_{L_E}(w) \cdot N_R(w)}{\sum_{w \in \mathcal{V}_R} N_R^2(w)} \\ (3.28) \quad &= \frac{\sum_{w \in \mathcal{V}} (N_L(w) - N_E(w)) \cdot N_R(w)}{\sum_{w \in \mathcal{V}_R} N_R^2(w)} \\ &= \frac{\sum_{w \in \mathcal{V}} N_L(w) \cdot N_R(w)}{\sum_{w \in \mathcal{V}_R} N_R^2(w)} - \frac{\sum_{w \in \mathcal{V}} \overbrace{N_E(w) \cdot N_R(w)}^{=0 \text{ if } w \notin \mathcal{V}_E}}{\sum_{w \in \mathcal{V}_R} N_R^2(w)} \\ &= \underbrace{\hat{\lambda}}_{= \hat{\lambda}} + \Delta \hat{\lambda} \end{aligned} \quad (3.29)$$

where

$$\Delta \hat{\lambda} = - \frac{\sum_{w \in \mathcal{V}_E} N_E(w) \cdot N_R(w)}{\sum_{w \in \mathcal{V}_R} N_R^2(w)} \quad (3.30)$$

Both $\hat{\lambda}$ and $\sum_{w \in \mathcal{V}_R} N_R^2(w)$ are already known from the evaluation of L , and the sum over \mathcal{V}_E is very short (has only few summands) as $|\mathcal{V}_E|$ is usually very small. Note that $\Delta \hat{\lambda}$ is always negative (removing an edge can only reduce the redundancy of a lattice). Furthermore, redundancies are always positive, hence $\Delta \hat{\lambda}$ is always negative when removing an edge.

$$\begin{aligned} 0 &\leq \hat{\lambda}' \\ \Leftrightarrow 0 &\leq \hat{\lambda} + \Delta \hat{\lambda} \end{aligned}$$

Therefore, we have

$$-\hat{\lambda} \leq \Delta \hat{\lambda} \leq 0 \quad (3.31)$$

when removing an edge from a lattice.

For $\hat{\sigma}^{2'}$, we have

$$\begin{aligned} \hat{\sigma}^{2'} &= \sum_{w \in \mathcal{V}} \left(\hat{\lambda}' \cdot N_R(w) - N_{L_E}(w) \right)^2 \\ &= \sum_{w \in \mathcal{V}} \left((\hat{\lambda} + \Delta \hat{\lambda}) \cdot N_R(w) - (N_L(w) - N_E(w)) \right)^2 \end{aligned}$$

$$\begin{aligned}
&= \sum_{w \in \mathcal{V}} \left(\underbrace{\hat{\lambda} \cdot N_R(w) - N_L(w)}_A + \underbrace{\Delta \hat{\lambda} \cdot N_R(w) + N_E(w)}_B \right)^2 \quad (3.32) \\
&= \sum_{w \in \mathcal{V}} A^2 + 2 \cdot AB + B^2 \\
&= \underbrace{\sum_{w \in \mathcal{V}} \left(\hat{\lambda} \cdot N_R(w) - N_L(w) \right)^2}_{\hat{\sigma}^2} + \sum_{w \in \mathcal{V}} (2A + B) \cdot \underbrace{B}_{0 \text{ if } w \notin \mathcal{V}_E \cup \mathcal{V}_R} \\
&= \hat{\sigma}^2 + \Delta \hat{\sigma}^2 \quad (3.33)
\end{aligned}$$

where

$$\Delta \hat{\sigma}^2 = \sum_{w \in \mathcal{V}_E \cup \mathcal{V}_R} (2A + B) \cdot B \quad (3.34)$$

with A and B as indicated in 3.32. Note that

$$\mathcal{V}_E \cup \mathcal{V}_R = \mathcal{V}_E \setminus \mathcal{V}_R \oplus \mathcal{V}_E \cap \mathcal{V}_R \oplus \mathcal{V}_R \setminus \mathcal{V}_E \quad (3.35)$$

is a disjoint partitioning of $\mathcal{V}_E \cup \mathcal{V}_R$ and B simplifies to $\Delta \hat{\lambda} \cdot N_R(w)$ for $w \notin \mathcal{V}_E$, so

$$\begin{aligned}
\sum_{w \in \mathcal{V}_R \setminus \mathcal{V}_E} (2AB + B^2) &= \sum_{w \in \mathcal{V}_R \setminus \mathcal{V}_E} \left(2A \cdot \Delta \hat{\lambda} \cdot N_R(w) + \Delta \hat{\lambda}^2 \cdot N_R^2(w) \right) \\
&= \underbrace{\Delta \hat{\lambda} \sum_{w \in \mathcal{V}_R} 2A \cdot N_R(w)}_C + \underbrace{\Delta \hat{\lambda}^2 \sum_{w \in \mathcal{V}_R} N_R^2(w)}_D - \\
&\quad \sum_{w \in \mathcal{V}_R \cap \mathcal{V}_E} \Delta \hat{\lambda} \cdot N_R(w) \cdot \left(2A + \Delta \hat{\lambda} \cdot N_R(w) \right) \quad (3.36)
\end{aligned}$$

Note that the first two sums (C, D) in equation 3.36 do not depend on the current edge and can be precomputed. Furthermore, $2AB + B^2$ simplifies to $(N_E(w) - 2N_L(w)) \cdot N_E(w)$ for $w \notin \mathcal{V}_R$, hence

$$\sum_{w \in \mathcal{V}_E \setminus \mathcal{V}_R} (2AB + B^2) = \sum_{w \in \mathcal{V}_E \setminus \mathcal{V}_R} (N_E(w) - 2N_L(w)) \cdot N_E(w) \quad (3.37)$$

Putting things together, we get

$$\begin{aligned}
\Delta \hat{\sigma}^2 &= \Delta \hat{\lambda} (C + \Delta \hat{\lambda} \cdot D) + \\
&\quad \sum_{w \in \mathcal{V}_R \cap \mathcal{V}_E} \left(2AB + B^2 - \left(2A \cdot \underbrace{\Delta \hat{\lambda} \cdot N_R(w)}_{B - N_E(w)} + \Delta \hat{\lambda}^2 \cdot N_R^2(w) \right) \right) + \\
&\quad \sum_{w \in \mathcal{V}_E \setminus \mathcal{V}_R} (N_E(w) - 2N_L(w)) \cdot N_E(w) \\
&= \Delta \hat{\lambda} (C + \Delta \hat{\lambda} \cdot D) + \\
&\quad \sum_{w \in \mathcal{V}_R \cap \mathcal{V}_E} \left(2AB - 2AB + 2A \cdot N_E(w) + B^2 - \Delta \hat{\lambda}^2 \cdot N_R^2(w) \right) +
\end{aligned}$$

$$\begin{aligned}
& \sum_{w \in \mathcal{V}_E \setminus \mathcal{V}_R} (N_E(w) - 2N_L(w)) \cdot N_E(w) \\
= & \Delta \hat{\lambda} (C + \Delta \hat{\lambda} \cdot D) + \\
& \sum_{w \in \mathcal{V}_R \cap \mathcal{V}_E} \left(2 \left(\hat{\lambda} N_R(w) - N_L(w) \right) \cdot N_E(w) + \Delta \hat{\lambda} \cdot N_R \cdot N_E(w) + N_E^2(w) \right) + \\
& \sum_{w \in \mathcal{V}_E \setminus \mathcal{V}_R} (N_E(w) - 2N_L(w)) \cdot N_E(w) \\
= & \Delta \hat{\lambda} (C + \Delta \hat{\lambda} \cdot D) + \\
& \sum_{w \in \mathcal{V}_E} \left[\delta_{w \in \mathcal{V}_R} \left(\left(2 \left(\hat{\lambda}' N_R(w) - N_L(w) \right) + N_E(w) \right) N_E(w) \right) \right. \\
& \left. + (1 - \delta_{w \in \mathcal{V}_R}) \left((N_E(w) - 2N_L(w)) \cdot N_E(w) \right) \right] \tag{3.38}
\end{aligned}$$

where

$$\delta_{w \in X} := \begin{cases} 1 & \text{if } w \in X \\ 0 & \text{otherwise} \end{cases} \tag{3.39}$$

and

$$C = \sum_{w \in \mathcal{V}_R} 2(\hat{\lambda} \cdot N_R - N_L) \cdot N_R(w) \tag{3.40}$$

$$D = \sum_{w \in \mathcal{V}_R} N_R^2(w) \tag{3.41}$$

Hence, calculation of $\Delta \hat{\sigma}^2$ only requires to sum over the edge's target words, everything else is a lattice-wide constant that can be precomputed, which is very fast.

3.4.2 The pruning criterion

We will prune an edge if, and only if, this will improve the lattice's total score with respect to the SWCD metric. Note that the lattice length $|L_i|$ and therefore also the lattice length sum $\sum |L_i|$ remains constant during the whole (edge) pruning process, so we can omit the normalization by $\sum |L_i|$ in eq. 3.19 when analyzing the change of the metric on a single lattice. The SWCD metric is a cost metric (the smaller the better), which leads to the following *pruning criterion*:

$$\begin{aligned}
\frac{\sqrt{\hat{\sigma}^{2'}}}{\hat{\lambda}'} &< \frac{\sqrt{\hat{\sigma}^2}}{\hat{\lambda}} \Leftrightarrow \tag{3.42} \\
\frac{\hat{\sigma}^{2'}}{\hat{\sigma}^2} &< \left(\frac{\hat{\lambda}'}{\hat{\lambda}} \right)^2 \Leftrightarrow
\end{aligned}$$

$$1 + \frac{\Delta \hat{\sigma}^2}{\hat{\sigma}^2} - \left(1 + \frac{\Delta \hat{\lambda}}{\hat{\lambda}} \right)^2 < 0 \tag{3.43}$$

In the following, we will discuss some special scenarios:

- $\hat{\lambda} = 0$ can only be observed if lattice and reference vocabularies are disjoint (see p. 58). In this case, all lattice edges should be considered trash (at least with respect to the reference).

Just as for the SWCD score itself, the pruning criterion (eq. 3.42) is not defined for $\hat{\lambda} = 0$. As $\Delta\hat{\lambda}$ becomes 0 in this case (see eq. 3.31) we can not even tell whether the pruning criterion will hold in the limit as the second fraction in eq. 3.43 reduces to $\frac{0}{0}$ resp. we compare infinity against infinity in eq. 3.42.

As we expect the pruning criterion to hold, we can define $\Delta\hat{\lambda}/\hat{\lambda} = -\infty$,¹⁶ so that eq. 3.43 simplifies to $-\infty < 0$,¹⁷ which always holds.

Alternatively, we can substitute $\hat{\lambda}$ by $\hat{\Delta}$ (see eq. 3.22) in equation 3.43, as we did for the SWCD score itself (see sec. 3.3.2). In this case, eq. 3.43 simplifies to

$$\Delta\hat{\sigma}^2 < 0 \quad (3.44)$$

Having both $\Delta\hat{\lambda}$ and $\hat{\lambda}$ being 0, eq. 3.34 (resp. eq. 3.38) simplifies to

$$\begin{aligned} \Delta\hat{\sigma}^2 &= \sum_{w \in \mathcal{V}_E \cup \mathcal{V}_R} (N_E(w) - 2N_L(w)) \cdot N_E(w) \\ &= \sum_{w \in \mathcal{V}_E} \underbrace{(N_E(w) - 2N_L(w))}_{<0} \cdot \underbrace{N_E(w)}_{>0} \\ &< 0 \end{aligned} \quad (3.45)$$

The inequalities in (3.45) follow from the fact that $N_E(w) \leq N_L(w) \forall w \in \mathcal{V}_L$ for any edge E in the lattice and $N_E(w) > 0$ for any $w \in \mathcal{V}_E$ by definition of \mathcal{V}_E . Hence, the pruning criterion always holds, as expected.

The second alternative has the benefit of distinguishing edges even if we cannot derive any statistical evidence from the reference: Shorter edges generally achieve a higher score than larger ones (which makes sense; an edge might emit a single word that does not occur in the reference by coincidence, an edge that emits 20 words that do not occur in the reference is just trash) and edges emitting words that are highly frequent in the lattice anyway ($N_L(w) \gg N_E(w)$) are also considered more disposable than edges emitting rare words ($N_L(w) \approx N_E(w)$).

- $\hat{\sigma}^2 = 0$ is only possible for a perfect lattice, where we would not want to prune any edge. In this case $\Delta\hat{\sigma}^2$ would be positive as an inequality similar to eq. 3.31 holds for $\Delta\hat{\sigma}^2$:

$$-\hat{\sigma}^2 \leq \Delta\hat{\sigma}^2 < \infty \quad (3.46)$$

(we can only subtract up to $\hat{\sigma}^2$ from $\hat{\sigma}^2$, as the squared error sum is always positive). This motivates us to define $\Delta\hat{\sigma}^2/\hat{\sigma}^2 = +\infty$,¹⁸ which would cause the pruning criterion to fail for any edge.

¹⁶The negative sign is motivated by the fact that the nominator is always negative while the denominator is always positive. Anyway, the sign will be lost due to the $(\cdot)^2$ operator in eq. 3.43.

¹⁷Note that $\hat{\lambda} = 0$ and $\hat{\sigma}^2 = 0$ are mutually exclusive if the reference is not empty.

¹⁸If an edge emits just all the words occurring in the reference(s), $\Delta\hat{\sigma}^2$ can actually become 0, which makes continuous completion of the fraction necessary. In that case $\Delta\hat{\lambda}$ would be -1 . Theoretically, having $\Delta\hat{\sigma}^2 = 0$, $\Delta\hat{\lambda}$ might even be a negative integer smaller than -1 , namely if an edge emits the reference words multiple times.

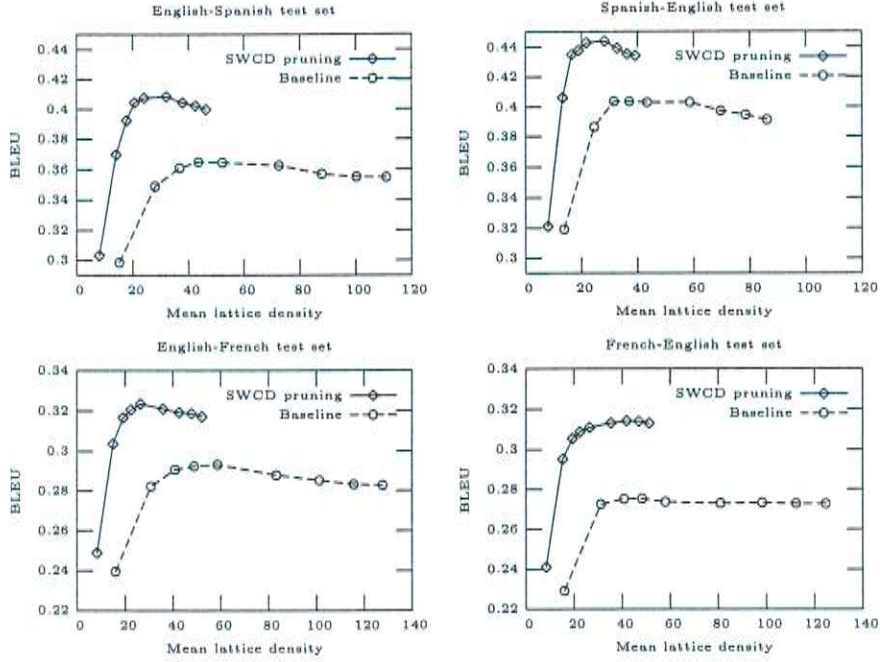


Figure 3.14: SWCD oracle pruning results in smaller lattices and better BLEU performance.

- If removing an edge that does not affect redundancy ($\Delta\hat{\lambda} = 0$), the pruning criterion reduces to $\Delta\hat{\sigma}^2 < 0$, that is, the edge removal would have to reduce the number of lattice-reference count mismatches.
- If the squared error sum does not change after edge removal ($\Delta\hat{\sigma}^2 = 0$), the pruning criterion simplifies to $\Delta\hat{\lambda} > 0$, which is purely hypothetical as the removal of an edge can only lower the redundancy (also see eq. 3.31).
- More generally, if the change of the squared error sum is positive ($\Delta\hat{\sigma}^2 > 0$) and the redundancy would decrease ($\Delta\hat{\lambda} < 0$) if the edge was removed, the pruning criterion cannot hold (the removal of the edge would worsen the lattice's quality by all means). On the other hand, if both squared error sum would decrease and redundancy would increase, the pruning criterion automatically holds. However, the latter is impossible as $\Delta\hat{\lambda} \leq 0$, so we can focus on edges for which $\Delta\hat{\sigma}^2$ (and $\Delta\hat{\lambda}$) would be negative.

We can focus on edges for which $\Delta\hat{\sigma}^2$ is negative, as the pruning criterion would automatically fail otherwise.

3.4.3 Decoder performance using SWCD pruning

Figure 3.14 shows experimental results on all four test sets. SWCD pruning on average discards between 44% and 58% of the edges of a lattice which leads to significantly smaller lattices.

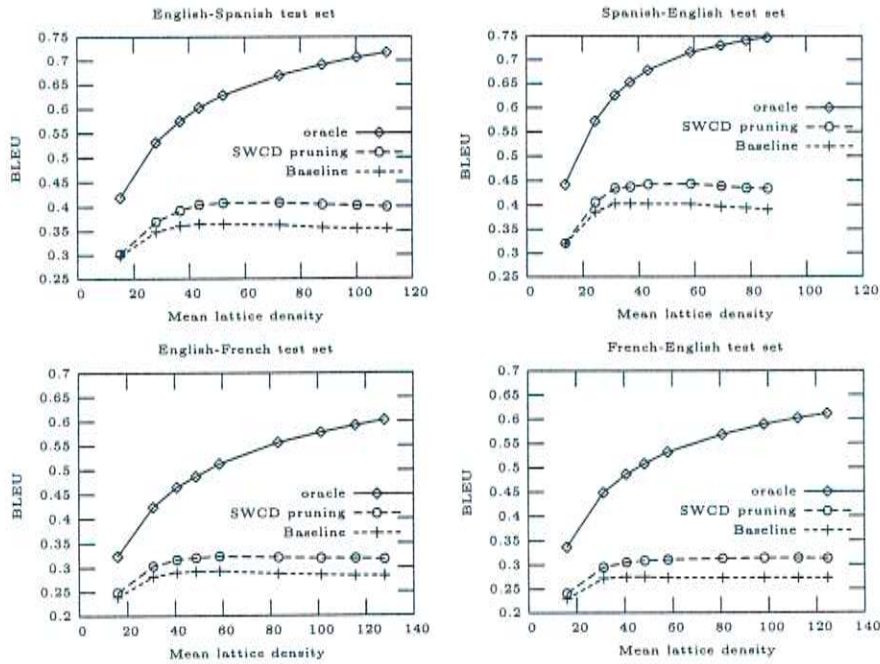


Figure 3.15: The oracle score of the Levenshtein decoder is approximately an upper bound for an “optimal” pruning algorithm. In this plot, we used the original (unpruned) lattice density to plot the SWCD pruning curve in order to make the three curves more comparable.

After SWCD pruning, the decoder outperforms the baseline system (comparing both optima) by about 10% to 14% (with respect to BLEU performance). This seems to be an impressive number, however, we have to keep in mind that this is an improvement using an *oracle* pruning method. A runtime pruning algorithm trained on the data resp. information provided by the SWCD pruning method could never fully exploit this improvement. Furthermore, we can use the BLEU performance of the Levenshtein decoder as an upper bound for an “optimal” pruning algorithm (or an optimal path search respectively). Simply speaking, the decoder will reach that upper bound if the pruning method manages to prune away all “bad” paths (except for those that are not erroneously found by the path search anyway). This is, of course, infeasible, an (oracle) pruning method that barely leaves more than the oracle path behind will probably be hopelessly overfitted.

Therefore, we do not expect the decoder to reach the upper bound given by the Levenshtein decoder using SWCD pruning, but comparing the plots of both systems on all four test sets (fig. 3.15), we see that there still is much space for improvement.

In general, there are two possible reasons why the oracle pruning method could not fully exploit the potential improvement indicated by the oracle performance: The first possibility is that the pruning method did not prune *enough* edges

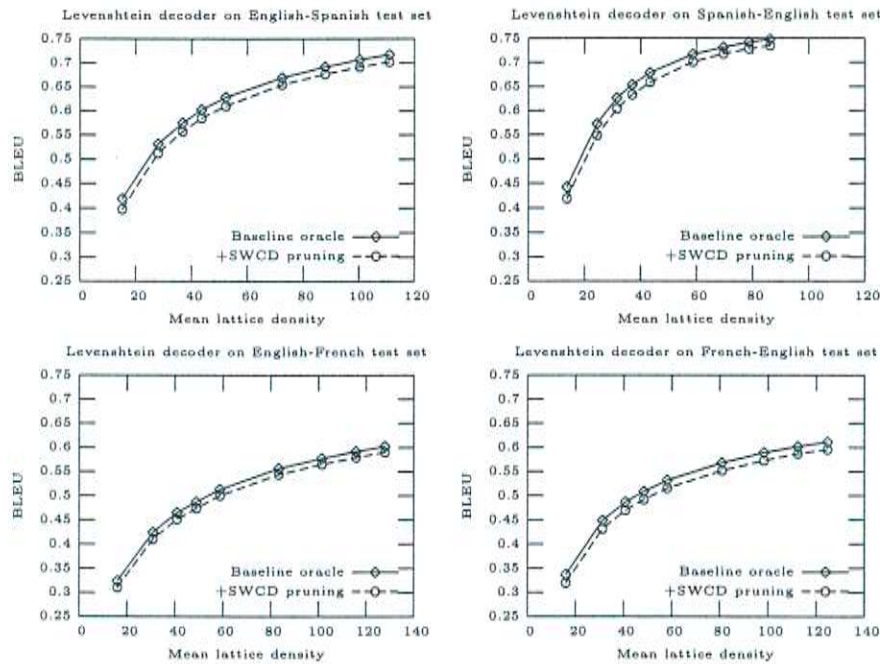


Figure 3.16: Errorously discarded edges due to SWCD pruning cause a constant drop on the Levenshtein decoder’s oracle score. As in fig. 3.15, the original lattice density was used to plot the SWCD pruning curve.

in order to force the path search to find more “optimal” (with respect to the reference) paths. On the other hand, the pruning method could have discarded the *wrong* edges. Any pruning technique will once in a while make a wrong decision by errorously discarding an edge of high quality. However, such an “error of first kind”¹⁹ should be in a due proportion to the improvements driven by the “right” decisions of low-quality edges being pruned away. We can meter the error of first kind by evaluating the pruned lattices using the Levenshtein decoder: Errors of first kind cause edges on the particular optimal paths to be lost, which leads to a drop in oracle performance. On average (depending on the test set), we lose between 1,3 and 1,8 BLEU points (out of 100) of oracle performance due to SWCD pruning. The drop is pretty constant, as we can see in figure 3.16. As the drop in oracle performance is reasonably small compared to the improvements on the end-to-end decoder performance using SWCD pruning, we conclude that we could prune much more progressively (which, however, increases the risk of overfitting).

¹⁹In decision theory and statistics, an *error of first kind* denotes a true (null) hypothesis being rejected. Furthermore, an *error of second kind* occurs if one accepts a false hypothesis. In our case, the (null) hypothesis is that the edge should stay in the lattice.

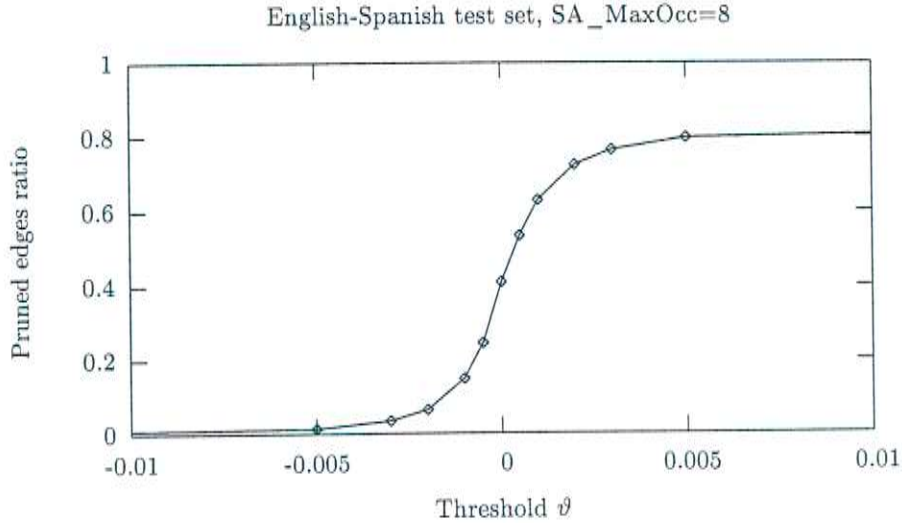


Figure 3.17: The relative amount of pruned edges with respect to the threshold ϑ shows properties of a *logistic growth function*. Logistic growth is a model of an exponential growth which is limited by a finite resource. Applications include the description of growth of a bacteria population or sales figures of a product in a mature market.

3.4.4 Introducing a threshold

Instead of pruning an edge as soon as we can observe a minimal decrease of the lattice's SWCD cost, we can compare the left-hand expression in eq. 3.43 against a threshold ϑ not necessarily being 0:

$$\underbrace{1 + \frac{\Delta\hat{\sigma}^2}{\hat{\sigma}^2} - \left(1 + \frac{\Delta\hat{\lambda}}{\hat{\lambda}}\right)^2}_{\tau} < \vartheta \quad (3.47)$$

We will write $\tau(E, L, R)$ to denote that the *test statistic* τ depends on the edge E , the lattice L and the reference(s) R , or simply $\tau(E)$ if we assume L and R to be fixed.

The more negative ϑ becomes, the more conservative the pruning will be. If ϑ is set to be slightly positive, edges are even pruned if it does not “hurt too much”. For $\vartheta = 0$, eq. 3.47 simplifies to eq. 3.43.

Figure 3.17 shows the relative amount of discarded edges with respect to ϑ . During the *growth phase* between $\vartheta = \pm 0.001$, the relative number of pruned edges grows approximately linearly, which enables us to easily control how strong our pruning criterion should be. The fact that the curve approximates 0 for $\vartheta \rightarrow -\infty$ more quickly than 1 for $\vartheta \rightarrow \infty$ indicates that the lattices a priori contains more “good” edges than “bad” ones.

The original pruning criterion does not change if eq. 3.43 is multiplied with or divided by a strictly positive constant c . For pruning with a threshold, this

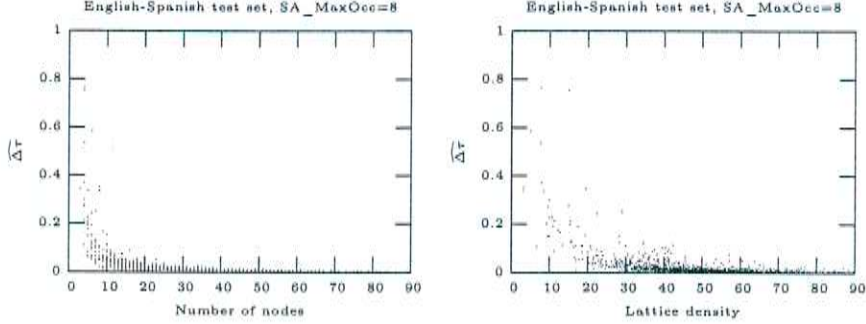


Figure 3.18: Lattice span $\widehat{\Delta\tau}$ with respect to the number of lattice nodes (**Left**) resp. lattice density (**Right**). The maximal number of source phrases for online phrase extraction (which is the decoder parameter that was used to vary the lattice density) was set to 8, leading to a mean lattice density of 46.7, which roughly hits the optimum in the decoder’s performance-density graph for the English-Spanish test set (see, for example, fig. 3.14, upper left).

means we can adapt the pruning criterion to the lattice by multiplying resp. dividing the threshold by a lattice-dependent constant. As said before, this does not change the original pruning criterion if $\vartheta = 0$.

In order to investigate whether the threshold ϑ should be adapted to the lattice, we consider the *span* of a lattice, the difference between the smallest and the biggest value of τ for all edges in a lattice $L = (\{n_0, \dots, n_k\}, \mathcal{E}, n_0, \{n_k\})$:

$$\widehat{\Delta\tau}(L) := \max_{E, E' \in \mathcal{E}} \tau(E) - \tau(E') \quad (3.48)$$

Our approach is to choose ϑ with respect to $\widehat{\Delta\tau}$. Having

$$\vartheta \sim \widehat{\Delta\tau} \quad (3.49)$$

we can expect to prune a constant relative amount of edges per lattice.

We correlated $\widehat{\Delta\tau}$ with some simple lattice-dependent constants for a fixed baseline system on the English-Spanish test set. Each dot in fig. 3.19 denotes the *span* of a lattice with respect to the number of edges. Analogously, we correlated $\widehat{\Delta\tau}$ with the number of lattice nodes respectively the lattice density in fig. 3.18.

All three plots approximate a hyperbolic curve, although the correlation is most evident with respect to the number of edges. Therefore, we assume that over the test set

$$\widehat{\Delta\tau}(L) \sim \frac{1}{|\mathcal{E}|} \quad (3.50)$$

$$\Leftrightarrow \widehat{\Delta\tau}(L) \cdot |\mathcal{E}| \approx \text{const} \quad (3.51)$$

This motivates us to consider

$$\tau'(E) := |\mathcal{E}| \cdot \tau(E) \quad (3.52)$$

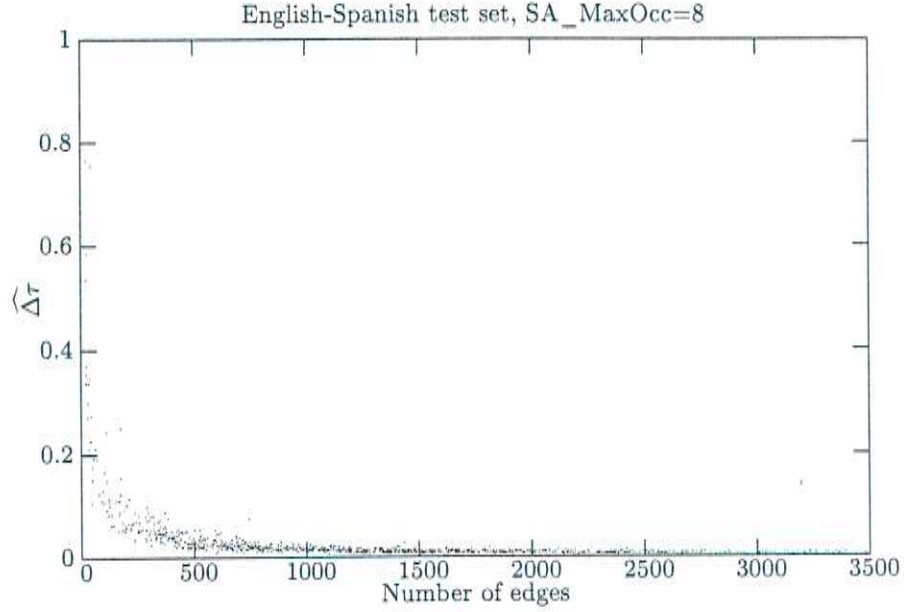


Figure 3.19: Lattice span with respect to the number of edges in the lattice. The dot cloud clearly approximates a hyperbolic curve.

instead of $\tau(E)$, because normalizing τ this way also normalizes $\widehat{\Delta\tau}$:

$$\begin{aligned}
 \widehat{\Delta\tau}'(L) &\stackrel{(3.48)}{=} \max_{E, E' \in \mathcal{E}} \tau'(E) - \tau'(E') \\
 &\stackrel{(3.52)}{=} \max_{E, E' \in \mathcal{E}} (|\mathcal{E}| \cdot \tau(E)) - (|\mathcal{E}| \cdot \tau(E')) \\
 &= |\mathcal{E}| \cdot \max_{E, E' \in \mathcal{E}} \tau(E) - \tau(E') \\
 &\stackrel{(3.48)}{=} |\mathcal{E}| \cdot \widehat{\Delta\tau} \\
 &\stackrel{(3.51)}{\approx} \text{const}
 \end{aligned}$$

which is what was wanted.

Using τ' instead of τ is equivalent to using $\vartheta/|\mathcal{E}|$ instead of ϑ for pruning:

$$\begin{aligned}
 \tau'(E) = |\mathcal{E}| \cdot \tau(E) < \vartheta &\Leftrightarrow \\
 \tau(E) < \vartheta' &
 \end{aligned} \tag{3.53}$$

where

$$\vartheta' := \frac{\vartheta}{|\mathcal{E}|} \tag{3.54}$$

is the *normalized threshold*. Hence, we have $\vartheta' \sim \widehat{\Delta\tau}$, that is, ϑ' fulfills eq. 3.49, which is what was wanted.

With ϑ being divided by a constant $|\mathcal{E}| \gg 1$, it is clear that we need much greater values for ϑ now. Figure 3.20 shows that for the relative amount of discarded edges, the phase of approximately linear growth now lies within $\vartheta = -2 \dots 2$.

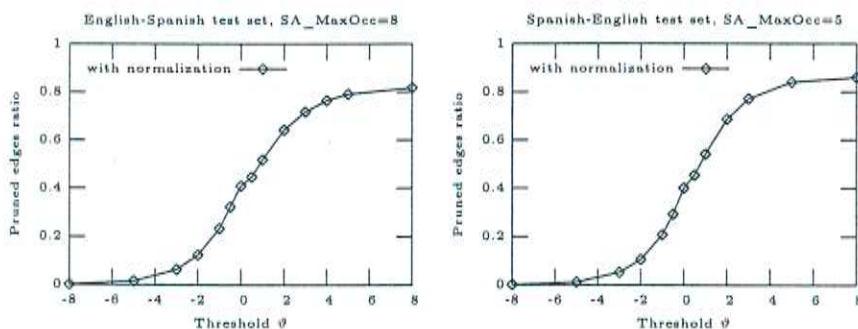


Figure 3.20: Using threshold normalization, the relative amount of pruned edges grows approximately linearly with respect to ϑ between $\vartheta = -2$ and $\vartheta = 2$. This observation holds for the English-Spanish test set (**Left**) as well as for the Spanish-English test set (**Right**, using a fixed baseline system with a mean lattice density of 31.5).

3.4.5 Decoder performance using SWCD threshold pruning

Figure 3.21 shows how the decoder’s end-to-end performance increases the more edges we prune. This is only natural as the pruning algorithm discards more and more “bad” paths. However, we have to keep in mind that using an oracle pruning method, we will eventually tend to *overfit* the (pruned) lattice to the reference. Differently speaking, if we use the oracle pruning decisions to provide training data to a runtime pruning technique, the trained algorithm will lose performance at some point if we (oracle) prune too much. As the optimal choice of ϑ depends on the used runtime pruning algorithm resp. training method etc., we cannot tell yet what would be reasonable values for ϑ .

Asymptotic behavior of ϑ

In fig. 3.21, we only considered values for ϑ that lie within the “phase of linear growth” with respect to the relative amount of pruned edges (see fig. 3.20). We will investigate the system’s behavior for $\vartheta \rightarrow \pm\infty$:

With decreasing ϑ , less and less edges will be pruned, so for very small thresholds ($\vartheta \ll -2$), the lattices are hardly modified. Hence, the system approaches the baseline system for $\vartheta \rightarrow -\infty$. In fig. 3.22, for example, the curve of the baseline system and the curve of the system using SWCD pruning cannot be distinguished for $\vartheta = -8$.

On the other hand, compared to the corresponding baseline system without SWCD pruning, the decoder’s end-to-end performance significantly drops if we choose an extremely high pruning threshold (e. g. $\vartheta = 8$, see fig. 3.22, left). Actually, it can be proven that the end-to-end performance drops on any test set if we just choose ϑ to be big enough:

PROOF As all lattices $L_i = (\{n_0^i, \dots, n_{k_i}^i\}, \mathcal{E}_i, n_0^i, \{n_{k_i}^i\})$ of the test

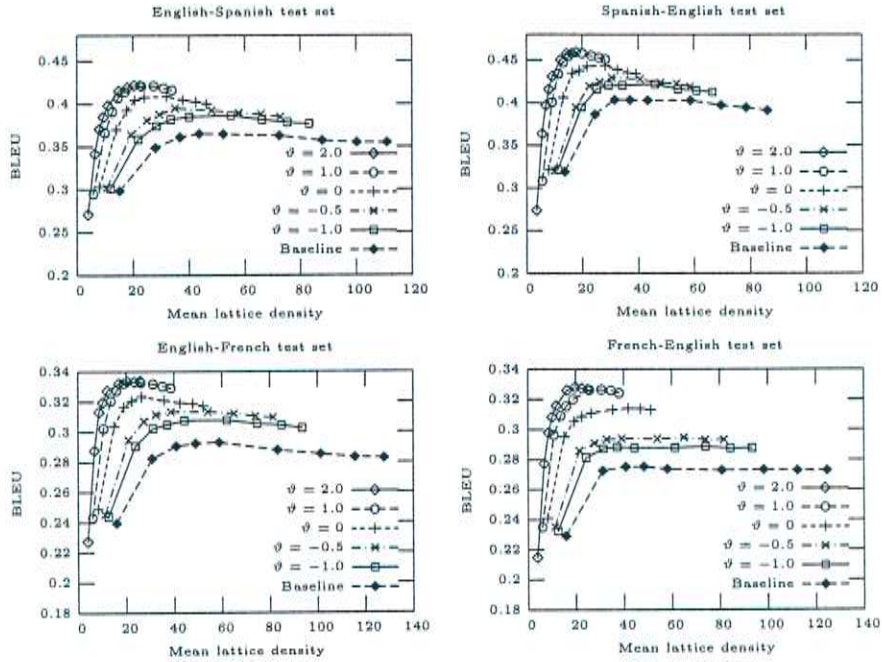


Figure 3.21: Nonsurprisingly, lattices become smaller with increasing threshold. Furthermore, the path search finds better paths with respect to BLEU.

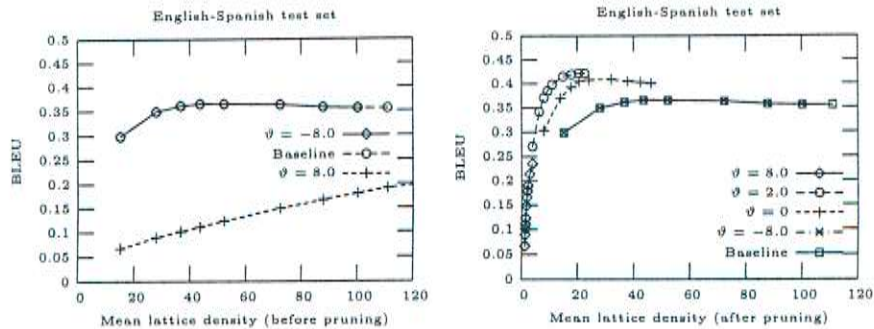


Figure 3.22: **(Left)** A high pruning threshold of $\vartheta = 8$ causes a significant drop of end-to-end decoder performance, compared to the corresponding baseline system. A very low threshold of $\vartheta = -8$ causes the system to perform virtually identically with the corresponding baseline system. **(Right)** If we take the reduced lattice density due to pruning into account, it becomes clear that the system using a high threshold might eventually outperform all other systems if we just construct a large enough unpruned lattice in the first place. It might also approximate the performance of the system having $\vartheta = 2$. Experiments in this direction could not be performed yet due to system limitations.

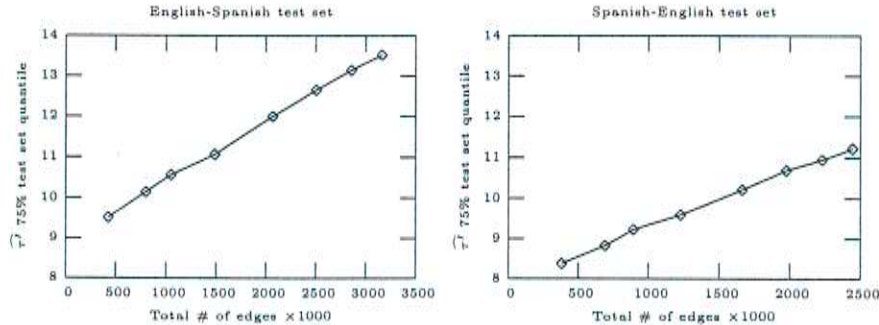


Figure 3.23: This plot shows the 75% quantile (which is robust against statistical mavericks) of the normalized test statistic $\hat{\tau}$ (see eq. 3.55) on the English-Spanish resp. Spanish-English test set, having the total number of lattice edges on the abscissa.

set $(L_i, R_i)_{i=1}^N$ contain a finite number of edges,

$$\hat{\tau}' := \max_{\substack{i=1, \dots, N \\ E \in \mathcal{E}_i}} \tau'(E, R_i) \quad (3.55)$$

exists (where τ' is the *normalized test statistic*, see eq. 3.52). Hence, if we set $\vartheta > \hat{\tau}'$, we can brutally discard all edges²⁰ which leads to a total information loss, that is, a minimal end-to-end performance of the decoder. ■

On the other hand, however, we can also observe a general tendency that the local maximum in the decoder's performance-density graph is shifted to the right (with respect to the *unpruned* lattice density) with increasing ϑ . Differently speaking, the stronger we prune, the more we can benefit from larger (unpruned) lattices. The effect becomes clearer if we plot the decoder's performance with respect to the density of the *pruned* lattice (fig. 3.22, right). The curve of the system using a big threshold has not reached its maximum yet, the corresponding system could still outperform all other systems if we manage to construct a big enough lattice in the first place that contains enough edges that pass the pruning criterion.

So far, it is unclear whether $\hat{\tau}'$ is limited or not. Figure 3.23 suggest that $\hat{\tau}'$ grows linearly with the number of lattice edges in the test set. In this case, $\hat{\tau}'$ would be unlimited and it would be possible to find arbitrarily many edges (resp. source-target phrase pairs) that pass the pruning criterion for virtually any choice of ϑ . However, it might also be the case that limiting effects will come into play for much larger lattices, which would mean that we could discard any edge as soon as we choose ϑ to be larger than that limit.

Anyway, as outlined before, we have to keep in mind that strong oracle pruning encourages side-effects due to overfitting, so we will focus on values for ϑ between -2 and 2 .

²⁰For all experiments, some edges were marked "unprunable" to make sure that there always exists a path through the lattice.

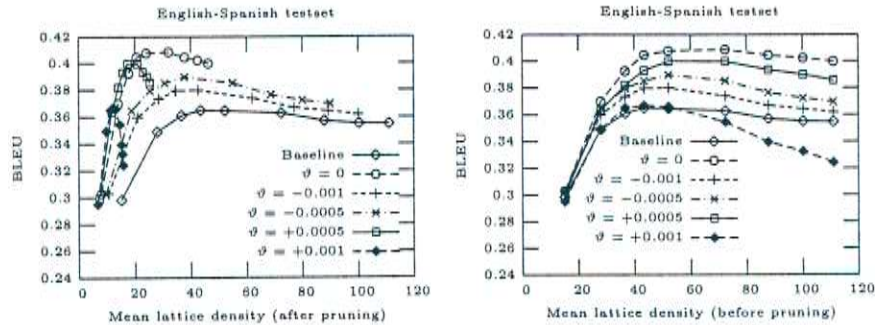


Figure 3.24: Decoder performance using SWCD pruning with unnormalized threshold on English-Spanish test set. Without threshold normalization (eq. 3.52), using a threshold only worsens the result (compared to $\vartheta = 0$).

Using an unadapted threshold

We also investigated what happens if we use thresholds directly (eq. 3.47), without normalization (eq. 3.52). Naturally, lattices still get smaller with increasing threshold. But figure 3.24 clearly shows that without threshold adaption, any value for $\vartheta \neq 0$ decreases the decoder performance (compared to $\vartheta = 0$), no matter whether we take the gains in smaller lattice density into account or not. For $\vartheta = +0.001$, the end-to-end performance of the decoder using SWCD pruning is even lower than the performance of the corresponding baseline system in some cases (not taking the smaller lattice density into account).

3.4.6 SWCD pruning compared to optimal pruning

Unfortunately, even using adaptive thresholds, SWCD pruning still does not seem to be able to fully exploit the potential gain a perfect pruning method could achieve, as indicated by the Levenshtein decoder’s oracle score (fig. 3.25). To us, no empirical results are known for this issue. It is well possible that the gain we observed using SWCD pruning is already pretty good for an oracle pruning method — except for the trivial possibility to prune away all edges that do not lie on the oracle path, which is, of course, infeasible. Furthermore, we have to keep in mind that lattices are much more compact after pruning, which allows for a more efficient path search. So far, we only applied SWCD pruning without changing path search parameters. However, also recall that lattice pruning also effects the lattice’s oracle score (see fig. 3.16), which is an indicator of how much “good” information has been lost due to pruning. If a pruning method manages to significantly reduce the size of a lattice without losing too much oracle performance, but we cannot observe a significant improvement of the decoder’s end-to-end performance anyway, we can assume that the path search did just not manage to exploit the benefit of a smaller lattice and search space. Conversely, if the pruning method causes a significant loss of oracle performance, even a perfect path search cannot make up for the lost information.

Figure 3.26 shows how SWCD pruning impacts the Levenshtein decoder’s oracle

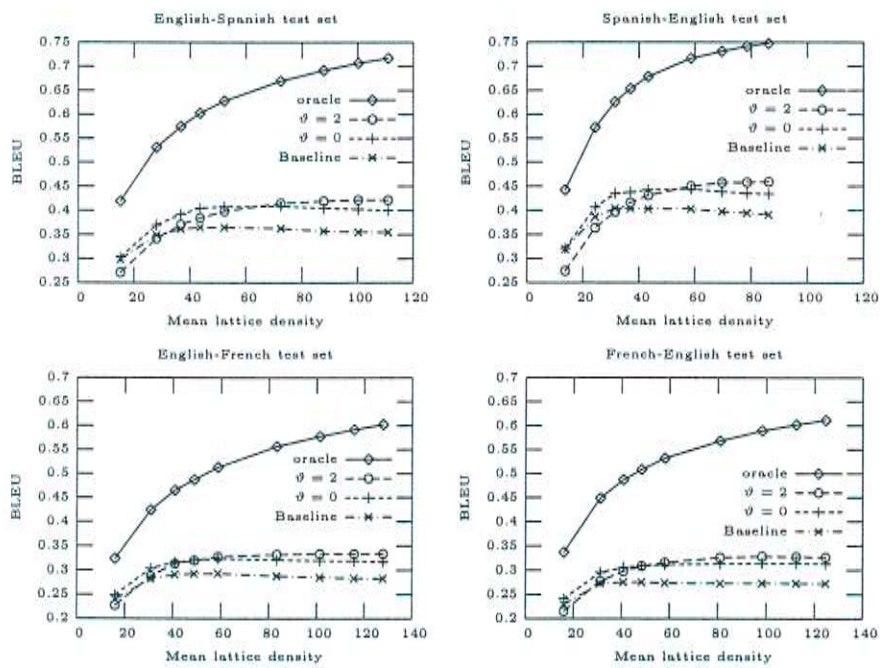


Figure 3.25: Baseline performance, oracle performance and decoder performance using SWCD pruning with or without threshold — all with respect to the lattice density *before* pruning (for direct comparability). Compared to the oracle performance, SWCD pruning only seems to bring minor improvements on the end-to-end decoder performance.

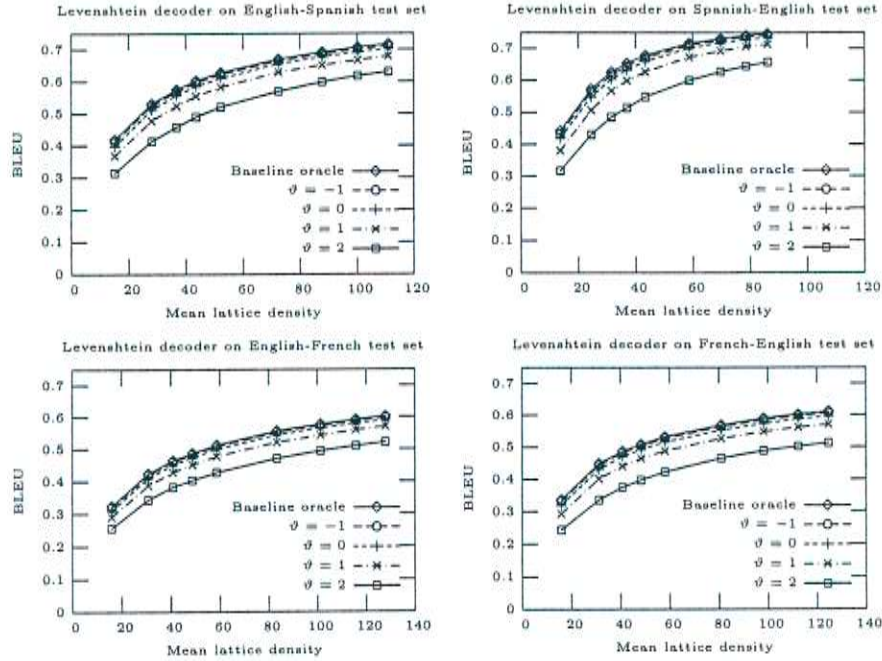


Figure 3.26: Levenshtein decoder oracle score using SWCD pruning with an adaptive threshold, with respect to the density of the unpruned lattice. The stronger we prune (that is, the bigger ϑ becomes), the greater is the loss on the oracle score, which indicates that we lose more and more “good” paths.

performance. We lose more oracle performance the more we prune, which was what was expected. In fig. 3.27, we can clearly see how the approaching upper bound forces the decoder using SWCD pruning ($\vartheta = 2$) to even drop below the baseline system’s performance.

However, this phenomenon effectively only affects the decoder for small lattice densities. For larger lattices, where the oracle score on the pruned lattice is still big enough, we can observe significant improvements. Differently speaking, as we have noticed before, the decoder reaches its local optimum for larger (unpruned lattices) with increasing ϑ .

On the other hand, if we consider oracle and decoder performance with respect to the density of the *pruned* lattice (fig. 3.28), the particular local maxima of the decoder systems drift to the *left* (also see fig. 3.21) and oracle and decoder score grow *apart* (or at least hardly lose distance) with increasing ϑ , considering a fixed (pruned) lattice density. This means that we reach (higher) optimal performance on smaller, more compact lattices and have at least the same potential for path search optimizations after pruning.

Comparing the decoder’s performance with the corresponding oracle score also helps to explain the performance drop for very large thresholds (see fig. 3.22). The decoder performance is so low because the (limiting) Levenshtein decoder

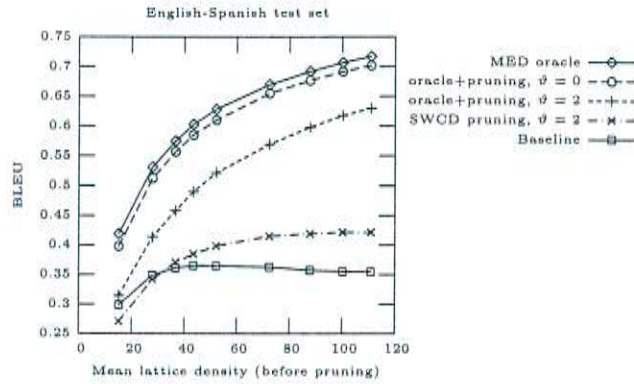


Figure 3.27: With stronger pruning, oracle score and decoder end-to-end performance approach each other. The decreasing oracle score limiting the decoder performance eventually causes the decoder’s performance to drop as well.

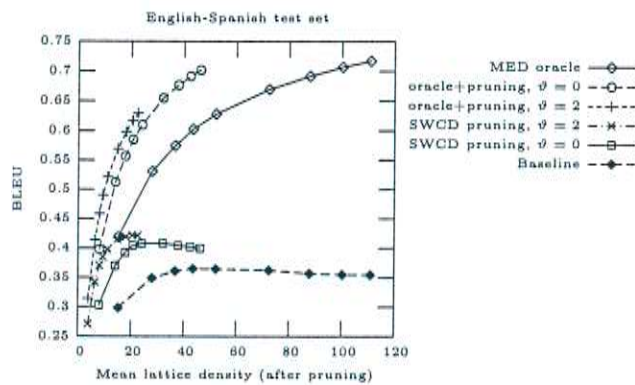


Figure 3.28: Decoder vs. oracle performance with respect to the density of the pruned lattice for increasing $\hat{\nu}$.

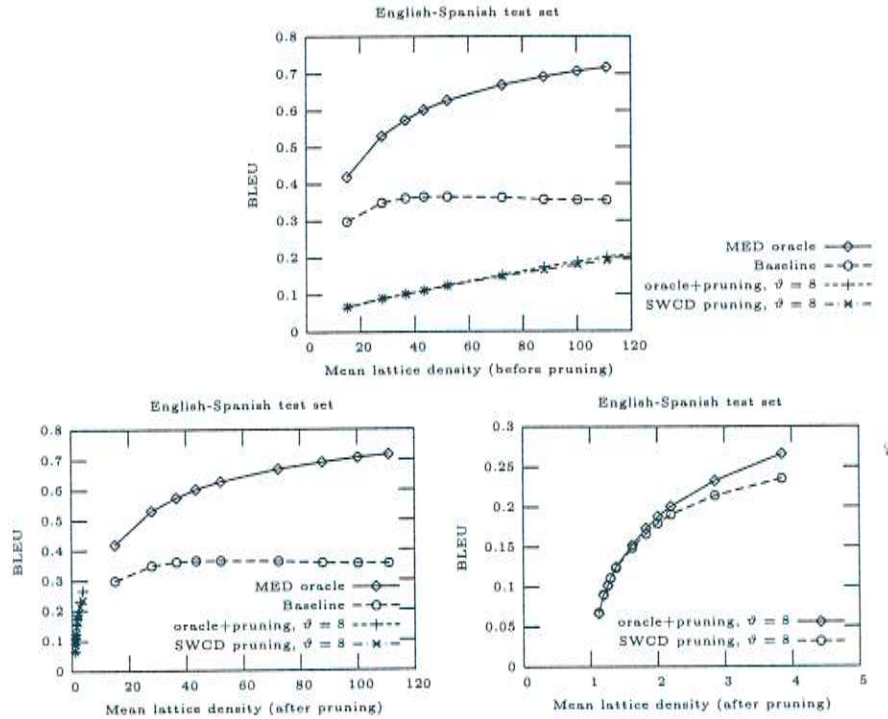


Figure 3.29: Oracle vs. decoder performance for large ψ : The decoder’s performance matching the oracle score indicates a perfect search. The plot on the bottom right shows an enlarged cutout of the plot to its left.

oracle score is already very low (fig. 3.29, top), due to the fact that the pruned lattice hardly contains “good” paths any more. Actually, the pruned lattices contain hardly paths *at all* (indicated by pruned lattice densities between 1 and 2.5, see fig. 3.29, bottom right), which also explains why the decoder’s performance matches the oracle score, indicating a perfect search.

3.5 Conclusions

aspects of translation lattice quality

In section 3.1, we investigated various aspects of *translation lattice quality*. In addition to *precision* and *recall*, the quality of a translation lattice also depends on its *uniformity* and its *compactness*.

We defined translation lattice quality in terms of end-to-end translation quality: One lattice is “better” than the other if, and only if, it leads to “better” translations.

We found that the topology of a translation lattice already contains some information about its quality. Furthermore, we found that oracle scores (see chapter 2) including the Graph Error Rate use the information contained in a translation lattice inefficiently. Therefore, we investigated novel approaches that exploit the full information contained in a translation lattice.

In section 3.3, we introduced a novel, model-free, unigram-based translation lattice evaluation metric called *Standard Word Count Distance (SWCD)* and investigated how it satisfies the various design criteria for translation lattice evaluation metrics stated on p. 54. We were able to show that Standard Word Count Distance is linearly correlated with the decoder’s end-to-end performance.

Standard Word Count
Distance

From SWCD, we derived an *oracle pruning* method based on how single edges contribute to the total cost of a lattice with respect to SWCD. We found an efficient way to calculate the test statistic used for the pruning criterion.

SWCD oracle pruning

On our test sets, we managed to reduce the size of the particular translation lattices significantly, accompanied by moderate improvements in end-to-end decoder performance without modification of the path search parameters. We used oracle scores to meter performance losses due to *errors of first kind*. We observed that the harder we prune, the more we can take advantage of larger lattices. However, we realized that the more we prune, the more we tend to *overfit* the pruned lattice to the corresponding reference.

errors of first kind

overfitting

Chapter 4

Conclusions

In the course of this thesis, we investigated various ways of translation lattice evaluation.

In chapter 2, we put our focus on *oracle decoding* which is an instance of the more general concept of *single-path evaluation*.

Two oracle decoders for two different objective functions (*Levenshtein distance* and *Position-independent Minimal Edit Distance (PMED)*) have been implemented. Although both forced aligners can be used for translation lattice evaluation tasks, they lack certain desirable features. Above all, as they rely on a single path extracted from the translation lattice, they use the information provided by the lattice unefficiently. They do not allow for a qualitative statement on non-path edges, that is, they do not help to discriminate between “good” and “bad” edges of a translation lattice. Last but not least, the correlation between oracle decoders and the end-to-end decoder performance highly depends on the lattice density.

In section 3.3, we introduced an approach completely different from oracle scores. The *Standard Word Count Distance (SWCD)* score evaluates lattices directly, without decoding. This way, various negative side-effects of single-path scores can be avoided, yet maintaining a high correlation with the decoder’s end-to-end performance.

From SWCD, we derived an *oracle pruning algorithm* for translation lattices. We found that the Levenshtein decoder’s oracle score can be used to meter the loss of information due to wrongfully pruned lattice edges. We managed to significantly reduce the size of our test set lattices accompanied by moderate improvements in end-to-end decoder performance.

4.1 Future Objectives

4.1.1 Model dependence

Throughout this thesis, we focused on *model-free* lattice evaluation. For future research, we suggest to also consider metrics that take the statistical models

model-dependent translation lattice evaluation metrics

(language model, word-to-word translation model etc.) into account. The main benefit of model-dependent translation lattice evaluation metrics is that they are more likely to be search-algorithm independent: We can significantly relax the s. a. i. criterion (def. 3.4) if we consider the statistical model(s) fixed for the set of path search algorithms for which eq. 3.6 should hold. Differently speaking, if we do not consider the scoring of lattices and paths to be part of the path search anymore, we can reduce the task of the path search to finding a path that is optimal *with respect to the scores* (so far we expected the path search to find a path that is optimal *with respect to the reference*), which is a much easier task; there is less “uncertainty” about the path search algorithm in eq. 3.6. If we assume that the path search at least approximately fulfills its specification, we can reduce the s. a. i. criterion to the criterion of *predicting decoder performance* for an optimal (with respect to the models) path search. Furthermore, we do not have to expect out statistical models to be consistent (def. 3.6) any more.

On the other hand, however, we have to make sure that the translation lattice evaluation metric verifies that the “good” pathes (with respect to the reference(s)) also have high scores resp. low costs (with respect to the statistical models). Basically, this makes the construction of model-dependent translation lattice evaluation metrics harder, but — as outlined before — model dependence also gives us some more flexibility.

4.1.2 Translation lattice optimization

Translation lattice evaluation metrics enable us to optimize parameters influencing the lattice generation process *directly*. This way one can decouple the optimization of the decoder’s lattice generation step from the optimization of the search algorithm (model weights, beam sizes, etc.).

We suggest that research effort should be put into how translation lattice evaluation metrics can help to optimize the translation lattice generation process. In particular, we recommend to use the results of this thesis in order to develop a runtime arbiter being part of CMU’s *Statistical Translation Toolkit* as a mediator between the phrase extraction and the translation lattice generation module¹. The arbiter should decide whether to overtake an edge resp. aligned phrase into the translation lattice or not. As opposed to *oracle pruning*, the arbiter must make its decisions based on previously trained parameters. The task of designing such an arbiter can be dissected into three parts:

- Constructing a *decision criterion*,
- selecting a *learning method* and
- providing *training data*.

decision criterion

The *decision criterion* also includes a selection of *features* of an edge resp. aligned phrase that should be taken into account. Standard approaches include the *perceptron* or — more generally — *neural networks*, but the *decision criterion*

¹Alternatively, we could generate a big lattice in the first place and then apply a post-pruning step. Such details are, however, rather technical.

can be any boolean expression based on the selected features. However, we have to keep in mind that a too specific design criterion might not be able to model the true decision boundaries, while design criteria which are too general tend to overfit and are harder to train.

The learning method heavily depends on the choice of the decision criterion. There exist well-known training algorithms, especially for the standard approaches mentioned above (e. g. *perceptron learning* for perceptrons or various variants of *backpropagation* for neural networks; see [Mit97] for details). Some decision criteria, however, require learning methods to be hand-tailored.

The learning method, in turn, implies what kind of training data is required. Some learning methods require a set of *positive and negative examples*. Others require a *ranking* of the provided examples or, more generally, a *quality function* which assigns a number to each example and which is to be learned directly.

The results of this thesis come into play for the last point. Using SWCD oracle pruning, we can provide positive and negative examples for edges that should or should not be discarded using the test statistic τ and the (normalized) threshold introduced in section 3.4.4. We can use two thresholds ϑ^+ and ϑ^- (where $\vartheta^- < \vartheta^+$ and $\{E \in \mathcal{E} \mid \tau(E) < \vartheta^-\}$ and $\{E \in \mathcal{E} \mid \tau(E) > \vartheta^+\}$ are the sets of negative resp. positive examples) to make sure that we keep the two training sets apart. The test statistic also provides a ranking of the edges, or we could try learn the test statistic directly, as a figure of merit.

We suggest that implementation and evaluation of such an arbiter should be subject to future research.

Bibliography

- [ABC⁺99] J.C. Amengual, J.M. Benedini, F. Casacuberta, A. Castaño, A. Castellanos, V.M. Jiménez, D. Llorens, A. Marzal, M. Pastor, F. Prat, E. Vidal, and J.M. Vilar. *The EUTRANS Speech Translation System*. Kluwer Academic Publishers, 1999.
- [Ata99] Mikhail J. Atallah, editor. *Algorithms and Theory of Computation Handbook*, pages 14–35. CRC Press LLC, 1999.
- [BCP⁺90] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2), June 1990.
- [BPP96] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–72, March 1996.
- [BPPM93] Peter Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June 1993.
- [BRJ99] Grady Booch, Jim Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley Longman, Inc., 1999.
- [CF04] Mauro Cettolo and Marcello Federico. Minimum Error Training of Log-Linear Translation Models. In *Proc. of the International Workshop on Spoken Language Translation*, pages 103–106, Kyoto, Japan, 2004.
- [CV04] F. Casacuberta and E. Vidal. Machine Translation with Inferred Stochastic Finite-State Transducers. 30(2):205–225, 2004.
- [Evg04] Evgeny Matusov and Maja Popović and Richard Zens and Hermann Ney. Statistical Machine Translation of Spontaneous Speech with Scarce Resources. In *Proc. of the International Workshop on Spoken Language Translation*, pages 139–146, Kyoto, Japan, 2004.
- [GC91] W. A. Gale and K. W. Church. A Program for Aligning Sentences in Bilingual Corpora. In *Proceedings of ACL-91*, Berkeley, CA, 1991.

- [Goo97a] Gerhard Goos. *Berechenbarkeit, formale Sprachen, Spezifikationen*, volume 3 of *Vorlesungen über Informatik*, chapter 15.1, “Reguläre Sprachen und endliche Automaten”. Springer-Verlag, 1997.
- [Goo97b] Gerhard Goos. *Grundlagen und funktionales Programmieren*, volume 1 of *Vorlesungen über Informatik*, chapter 1.6.3, “Chomsky-Grammatiken”. Springer-Verlag, second edition, 1997.
- [IDM03] Joseph P. Turian I. Dan Melamed, Ryan Green. Precision and Recall of Machine Translation. Proteus technical report #03-004, Computer Science Department, New York University, 2003.
- [Kau02] Manuel Kauers. Verstehen natürlicher Sprache durch statistische Übersetzung in eine termbasierte Interlingua. Diplomarbeit, Universität Karlsruhe (TH), May 2002.
- [Kni99] Kevin Knight. A Statistical MT Tutorial Workbook. Prepared in connection with the JHU summer workshop, April 30 1999.
- [Köh] Phillip Köhn. Europarl: A Multilingual Corpus for Evaluation of Machine Translation. Available at <http://people.csail.mit.edu/koehn/publications/europarl/>.
- [Kol30] A. N. Kolmogorov. Sur la loi forte des grands nombres. *Comptes Rendus de l'Academie des Sciences, Paris*, 191:910–912, 1930.
- [KVM⁺05] Stephan Kanthak, David Vilar, Evgeny Matusov, Richard Zens, and Hermann Ney. Novel Reordering Approaches in Phrase-Based Statistical Machine Translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 167–174, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.
- [LDC] LDC — Linguistic Data Consortium. Homepage at <http://www ldc.upenn.edu/>.
- [Lev02] Vladimir I. Levenshtein. Contributors. In *IEEE Transactions On Information Theory*, volume 48(7), pages 2143–2150. IEEE, July 2002.
- [LLL05] Yang Liu, Qun Liu, and Shouxun Lin. Log-linear models for word alignment. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 459–466, Ann Arbor, June 2005.
- [Mit97] Tom Michael Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [MS99] Christopher D. Manning and Heinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.
- [NIS] NIST MT evaluation kit version 11. WWW. Available at: <http://www.nist.gov/speech/tests/mt/>.
- [NOV00] Hermann Ney, Franz Josef Och, and Stephan Vogel. The RWTH System for Statistical Translation of Spoken Dialogues. In *Proceedings of the first international conference on Human Language Technology research*, pages 1–7, San Diego, 2000.

- [Och02] Franz Josef Och. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 2002.
- [Och03] Franz Josef Och. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of the ACL*, pages 160–167, Sapporo, Japan, 2003.
- [Oli04] Oliver Bender and Richard Zens and Evgeny Matusov and Hermann Ney. Alignment Templates: the RWTH SMT System. In *Proc. of the International Workshop on Spoken Language Translation*, pages 79–84, Kyoto, Japan, 2004.
- [ON00] Franz Josef Och and Hermann Ney. Improved Statistical Alignment Models. In *Proc. of the 39th Annual Meeting of the ACL*, 2000.
- [ON02] Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, July 2002.
- [PRWZ02] K.A. Papineni, S. Roukos, T. Ward, and W.J. Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, July 2002.
- [RW04] Jürgen Reichert and Alex Waibel. The ISL EDTRL System. In *Proc. of the International Workshop on Spoken Language Translation*, pages 61–64, Kyoto, Japan, 2004.
- [Sed92] Robert Sedgewick. *Algorithms in C++*. Addison-Wesley Publishing Company, Inc., 1992.
- [Sha48] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3-4):379–423, 623–656, July/October 1948.
- [Sha49] Claude E. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949.
- [TCS] TC-STAR: Technology and corpora for speech-to-speech translation. WWW. <http://www.tc-star.org>.
- [TIM02] Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. Extensions to HMM-based Statistical Word Alignment Models. In *Proc. Conf. on Empirical Methods for Natural Language Processing (EMNLP)*, pages 87–94, Philadelphia, PA, July 2002.
- [TL] SRI Speech Technology and Research Laboratory. SRILM — The SRI Language Modeling Toolkit. Available at <http://www.speech.sri.com/projects/srilm/>.

- [TN00] C. Tillmann and H. Ney. Word re-ordering and DP-based search in statistical machine translation. In *COLING '00: The 18th Int. Conf. on Computational Linguistics*, pages 850–856, Saarbrücken, Germany, August 2000.
- [TSM03] Joseph P. Turian, Luke Shen, and I. Dan Melamed. Evaluation of Machine Translation and its Evaluation. In *Machine Translation Summit IX*. International Association for Machine Translation, September 2003.
- [UN04] Nicola Ueffing and Hermann Ney. Bayes Decision Rules and Confidence Measures for Statistical Machine Translation. In *EsTAL-España for for Natural Language Processing*, pages 70–81, Alicante, Spain, October 2004. Lecture Notes in Computer Science, Springer Verlag.
- [UON02] Nicola Ueffing, Franz Josef Och, and Hermann Ney. Generation of Word Graphs in Statistical Machine Translation. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 156–163, Philadelphia, PA, July 2002.
- [V⁺] Stephan Vogel et al. The Statistical Translation Toolkit (STTK). Carnegie Mellon University, Pittsburgh, PA.
- [VHKW04] Stephan Vogel, Sanjika Hewavitharana, Muntsin Kolss, and Alex Waibel. The ISL Statistical Translation System for Spoken Language Translation. In *Proc. of the International Workshop on Spoken Language Translation*, pages 65–72, Kyoto, Japan, 2004.
- [VNT96] Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based Word Alignment in Statistical Translation. In *COLING '96: The 16th Int. Conf. on Computational Linguistics*, pages 836–841, Copenhagen, August 1996.
- [VON⁺00] Stephan Vogel, Franz Josef Och, S. Nießen, H. Sawaf, C. Tillmann, and Hermann Ney. *Statistical Methods for Machine Translation*, pages 377–393. Springer Verlag, Berlin, July 2000.
- [VZH⁺03] Stephan Vogel, Y. Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. The CMU Statistical Machine Translation System. In *Proc. of MT Summit IX*, New Orleans, LA, 2003.
- [Wika] Wikipedia. WWW. <http://www.wikipedia.org>.
- [Wikb] Wikipedia (deutsch). WWW. <http://de.wikipedia.org>.
- [Wu83] C. F. J. Wu. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11:95–103, 1983.
- [WW97] Ye-Yi Wang and Alex Waibel. Decoding Algorithm in Statistical Machine Translation. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of*

- the European Chapter of the Association for Computational Linguistics*, pages 366–372, Somerset, New Jersey, 1997. Association for Computational Linguistics.
- [ZN05] Richard Zens and Hermann Ney. Word Graphs for Statistical Machine Translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 191–198, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.
- [ZON02] Richard Zens, Franz Josef Och, and Hermann Ney. Phrase-based statistical machine translation. In *Proc. of the German Conference on Artificial Intelligence (KI 2002)*. Springer Verlag, September 2002.
- [ZV02] Bing Zhao and Stephan Vogel. Adaptive Parallel Sentences Mining from Web Bilingual News Collection. In *IEEE International Conference on Data Mining (ICDM) 02*, volume 00, pages 745–748, Japan, 2002.
- [ZV03] Bing Zhao and Stephan Vogel. Word alignment based on bilingual bracketing. In Rada Mihalcea and Ted Pedersen, editors, *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 15–18, Edmonton, Alberta, Canada, May 31 2003. Association for Computational Linguistics.
- [ZZVW03] Bing Zhao, Klaus Zechner, Stephen Vogel, and Alex Waibel. Efficient Optimization for Bilingual Sentence Alignment Based on Linear Regression. In Rada Mihalcea and Ted Pedersen, editors, *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 81–87, Edmonton, Alberta, Canada, May 31 2003. Association for Computational Linguistics.

Index

- $\Delta\hat{\lambda}$, 67
- $\Delta\hat{\sigma}^2$, 68
- $\Delta\tau$, *see* span of a lattice
- τ , *see* test statistic
- n -gram, 10, 51, 52, 54, 64
- alignment, 7
 - diagram, 8
- analysis, 2
- backpointer
 - in dynamic programming, 21
- bilingual bracketing, 8
- BLEU, 14
- bound lattice complexity, 45
- bracketing, bilingual, 8
- brevity penalty, 14
- canonical metric, *see* translation lattice, evaluation metric, canonical
- compactness, 43, 45, 54, 84
- concept, 1
- cost function, 45
- count, *see* word count
- data sparseness, 52
- decoder, ix, 10, 41
 - oracle \sim , *see* oracle decoder
- decoding, 6, 10
- deletion error, 20
- design criteria
 - for lattice evaluation metrics, 54
- direct translation, 2
- dynamic programming (DP), 20
- Edge Precision, 51
- EM algorithm, 8, 51
- EPPS, 15
- error
 - of first kind, 73, 73
 - of second kind, 72, 73
 - error of first kind, 85
 - evaluation
 - of translation systems, 14
 - evaluation function, 17
 - event, 4, 54
 - Expectation-Maximization, *see* EM algorithm
 - fan-out, 45
 - fluency, 19, 64
 - forced aligner, *see* oracle decoder, 17
 - forced alignment, 17
 - full lattice, 41
 - GBLEU, 19
 - generation, 1
 - generation function
 - deterministic, 2
 - indeterministic, 5
 - GPER, 19
 - Graph Error Rate, 18, 41, 50, 84
 - Graph Word Error Rate, 18
 - GWER, *see* Graph Word Error Rate
 - hypothesis
 - complete, 22
 - expansion, 22
 - final, 22
 - recombination, 22
 - insertion error, 20
 - interACT, vii
 - interlingua-based translation, 2
 - interpretation, 1
 - interpretation function
 - deterministic, 2
 - indeterministic, 5
 - knowledge-based translation, ix
 - language model, 7, 8, 10, 15
 - lattice

- complexity, 31, 45
- density, 31, 40, 45, 47
- evaluation metric, *see* translation lattice, evaluation metric
- full \sim , 41
- perfect \sim , 41
- quality, *see* translation lattice, quality
- size, 45
- lattice generation, 11
- lattice redundancy, 55
- lattice state of a hypothesis, 22
- Levenshtein
 - decoder, 17
 - distance, *see* Minimal Edit Distance
 - path, 21
 - Vladimir I., 17
- logistic growth, 74
- machine translation, ix
 - statistical \sim , 3
- MAP-classifier, 4
- maximum entropy, 9
- MED, *see* Minimal Edit Distance
- Minimal Edit Distance, 17, 39, 87
 - Position-independent \sim , 17
- minimum error rate training, 50
- model consistency, 47
- model error, 29
- model independence, 54
- multiple references, 52
- NIST, 14
- noisy channel, 6, 7
- objective function, 17
- Occam, *see* Ockham
- Ockham
 - Occam's Razor, 43
 - William of, 43
- oracle
 - decoder, x, 17, 66
 - decoding, 17, 87
 - experiment, 17
 - hypothesis, 17
 - pruning, 66, 85
 - pruning criterion, 69
 - score, 18, 84
- overfitting, 54, 73, 77, 85
- parametric learning, 33
- part-of-speech tag, 8
- path search, 11
 - pruning, 42
- PER, *see* Word Error Rate
- perfect lattice, 41
- perfect path, 41
- Performance-Density Graph, 31
- performance-density graph, 32, 47
- PMED, *see* Position-independent Minimal Edit Distance
- Position-independent Minimal Edit Distance, 19, 39, 87
- Position-independent Word Error Rate, 18
- precision, 14, 42, 44, 44, 54, 63, 84
- predict decoder performance, 46, 54
 - for model-independent metrics, 46
- pruning, 42, 47, 48, 50
 - oracle \sim , 66
 - oracle \sim criterion, 69
- pruning criterion, 85
- PWER, *see* Position-independent Word Error Rate
- quality
 - of a translation lattice, *see* translation lattice, quality
- recall, 14, 42, 44, 44, 54, 63, 84
- reordering, 13
 - global, 13
 - local, 13
- s.a.i., *see* search algorithm independence
- search algorithm independence, 45, 46
 - for model-independent metrics, 47
- search error, 30
- sentence length model, 9
- single path evaluation, 51
- single-path evaluation, 87
- smoothing, 9, 10
- span of a lattice, 75
- SRI toolkit, 14
- Standard Word Count Distance, 41, 54, 56, 85
- standard word count distance, x, 87
- statistical machine translation, ix
 - fundamental theorem of \sim , 4
- strong law of large numbers, 4
- STTK, 14, 17

- substitution error, 21
- SWCD, *see* Standard Word Count Distance
- TC-STAR, 15
- test events, 44
- test statistic, 74, 85
 - normalized, 75
- threshold, 74
 - normalized, 76
- topology, 41
- training corpora, 15
- transducer, *see* stochastic finite-state
 - \sim , 11
- transfer, 2
- transfer-based translation, 2
- translation lattice, ix, 11, 41
 - evaluation, 41
 - evaluation metric, 43
 - canonical, 48, 50
 - model-independent, 46
 - oracle pruning, 87
 - quality, x, 41, 84
 - raw, *see* translation lattice w/o edge weights
 - without edge weights, 43
- translation model, 4, 7
 - reverse, 7
- translation triangle, 2
- uniformity, 43, 45, 54, 63, 84
- unprunable edges, 79
- Viterbi alignment, 8
- WER, *see* Word Error Rate
- word counts, 9, 28
 - for lattice target words, 55
 - for multiple references, 57
- Word Error Rate, 17, 18
 - Graph \sim , *see* Graph Word Error Rate
- word graph, 11