Shallow Statistical Parsing for Machine Translation

Flache statistische Analyse in der maschinellen Übersetzung

Diplomarbeit

Universität Karlsruhe
Fakultät für Informatik

Marcus Munk

Betreuer:
Prof. A. Waibel

Betreuender Mitarbeiter:
Dipl.-Inform. Klaus Ries

31. Mai 1999

Zusammenfassung:

Die vorliegende Arbeit beschäftigt sich mit einem wahrscheinlichkeitstheoretischen Ansatz, der es ermöglicht mit Hilfe von statistischen Inferenztechniken Modellparameter automatisch von einem gegebenen Corpus (d.i. eine Sammlung von Sätzen) zu extrahieren. Der für das Training benötigte Corpus ist relativ klein (ca. 6.000 Sätze).

Im Kontext des C-STAR Projektes wurde ein System-Prototyp erstellt (SALT). C-STAR bezeichnet ein Konsortium zur wissenschaftlichen Kooperation auf dem Gebiet der Sprachübersetzung in der Reisebuchungsdomäne. Die Implementierung des Prototyps erfolgte unter Verwendung diverser Module der Janus Bibliothek, sowie weiterer Standardmodule zur Sprachverarbeitung. Das System besteht aus zwei Hauptkomponenten: ein Modul zur flachen statistischen Analyse, welches die Eingabesätze in Segmente zerteilt, und ein grammatikbasierter Parser, der dann die Segmente analysiert.

Um die Lernfähigkeit und Robustheit des Ansatzes zu demonstrieren wurde das SALT-System mit per Hand transkribierten und vom Parser annotierten Sätzen trainiert. Zur Evaluation wurden vom Spracherkenner erkannte Sätze wie auch transkribierte Sätze verwendet. Das System erzielt gute Resultate im Vergleich mit dem grammatikbasierten Parser, wenn man den prototypenhaften Zustand des Systems berücksichtigt.

Obwohl die derzeitige Leistung des SALT Systems nicht an den Parser anknüpfen kann, erzielt eine Kombination beider Systeme eine Verbesserung der Übersetzungsleistung, die beide Einzelsysteme übertrifft.

Das SALT-System kann mit geringem Aufwand in eine andere Domäne oder Sprache übertragen werden, was den hier vorgestellten Ansatz sehr wirtschaftlich erscheinen läßt.

Weitere Vorteile des vorgestellten Ansatzes sind die, dem statistischen Modell zugrunde liegende Robustheit, die Allgemeinheit und Wirtschaftlichkeit der Implementierung, da hauptsächlich domänen- und sprachunabhängige Module verwendet werden, sowie die Erweiterbarkeit des statistischen Modells, in dem sich relativ einfach weitere Informationsquellen, wie z.B. Prosodie, oder Lippenlesen integrieren lassen.

Die vorliegende Arbeit wurde als Diplomarbeit angefertigt an der

Universität Karlsruhe
Fakultät für Informatik
Institut für Logik, Komplexität
und Deduktionssysteme
Postfach 6980
D-76128 Karlsruhe

Der Betreuer der Arbeit war

Prof. A. Waibel

Erklärung:

Hiermit erkläre ich, die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Marcus Munk

Pittsburgh, den 31.05.1999

Abstract:

This work proposes an approach for shallow parsing of input utterances using a probabilistic framework, which allows derivation of model parameters automatically from a corpus, using statistical inference techniques. The required corpus for training the statistical models is relatively small (approximately 6,000 utterances).

A prototypical system (SALT) was developed and tested in the context of the C-STAR II machine translation effort, a research consortium cooperating on speech translation for travel planning dialogues. It was implemented using modules from the Janus RTK, as well as other standard natural language processing components. It consists of two main components: a shallow statistical analysis for detecting segments in input utterances, and a grammar-based parser for parsing the detected segments.

To demonstrate learn-ability and robustness SALT is trained with transcribed and automatically (by a grammar-based parser) annotated sentences, and evaluated on both, transcribed and recognized, sentences. It performs relatively well compared to a grammar-based parser, which has seen a considerable effort in development and grammar-writing.

Although SALT's current performance does not reach the level of the grammar-based parser, experiments integrating both systems into a multi-engine translation system state an improved performance, superior over both single systems.

The described approach is highly economical in porting. Given a sufficient amount of annotated training data SALT can be ported to a different domain or language with little effort, since almost no hand modelling is needed.

Further advantages of the proposed approach lie in its robustness, which is inherent in the statistical models; its general and economic implementation, which is due to the use of mostly language and domain independent standard components; and its extensibility, since the statistical models facilitate a simple integration of additional information, such as e.g. prosody, lip-reading, etc.

Acknowledgements:

This work would not have been possible without the help of a lot of folks. I can't possibly name everyone here, and undoubtedly I've overlooked at least one major contributor; but there are at least some folks that I'd like to thank publicly and profusely for their help and support:

First, I want to thank my parents for supporting me all the time during my studies; especially during the times I spent at UMASS and CMU.

Second, I would like to thank my advisors Prof. A. Waibel and Klaus Ries for their help and support. Especially, to Klaus I owe many insights into a field of research I was illiterate in before.

Third and last, I thank Chad Langley for proof-reading this work, and for helping me out in many other occasions as an exemplary office-mate.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The most common way that people communicate is by speaking or writing in one of the 'natural' languages, like English, German, French, Italian, Japanese, or Korean. The underlying assumption here is that the participants speak and understand the same common language. Otherwise a translator or interpreter would be needed, helping to cross language barriers — if one does not want to fall back on more fundamental ways of communication, e.g. such as drawings, gestures or facial expressions. In the background of world-spanning industries, international politics, as well as global tourism, the need for translation from one language into another becomes even more relevant.

Nevertheless, most translation in the world is not of texts which have high literary and cultural status. The great majority of translators are employed to satisfy the huge and growing demand for translations of scientific and technical documents, commercial and business transactions, administrative and legal documents, newspaper reports, etc. Some of this work is challenging and difficult; but much of it is tedious and repetitive, while at the same time requiring accuracy and consistency, and therefore demanding a 'mechanization' of translation. The assistance of a computer has clear and immediate attractions, ranging from intelligently automated dictionaries to fully automatic machine translation systems.

1.1 Machine Translation

The term *Machine Translation* (MT) is the now traditional and standard name for computerized systems responsible for the production of translations from one natural language into another, with or without human assistance. It has roots several hundred years back in the history of mankind: already Descartes noticed that the same words or concepts existed in the languages he knew and he introduced the motion of a mechanical dictionary, containing equivalents in several languages. However, serious and quite detailed technical proposals for MT were first put forward in 1933, when the French G. Artsruni and the Russian P. P. Smirnov-Trojanskij independently took out patents for the translation machines they had invented.

Methodology

In retrospect, it is worth having a closer look at Trojanskij's work, since it incorporated some of the design features and principles of MT that are still valid today. He recognized three stages in the translation process, which in modern terminology, could be described as *analysis*, *transfer*, and *generation*.

In analysis, the source language would be transformed into a canonical form A, which does not readily contain the information G that the target language generator needs; in transfer the source language canonical form A would then be mapped to the corresponding target language canonical form G, which means that the mapping $A \rightarrow G$ must be defined for each pair of languages; lastly, in generation, the target language output would be reconstructed.

Apart from the *Transfer Principle*, Trojanskij also fathered the notion of an 'intermediary language' or *interlingua*. He maintained that independently of their individual lexical and grammatical forms, all languages had a common logical content which allowed translation from one language to any other language via a universal logical intermediary language. The so-called *Interlingua Principle* is mostly applicable in small domains with a limited set of concepts, whereas the Transfer Principle is rather applied to large domains. In a domain with n languages the Interlingua Principle requires only n parsers and n generators, whereas the Transfer Principle needs $(n-1)^2$ transfer components additionally.

The first approach that was actually implemented in several computer projects is generally referred to as the *Direct Translation Principle*. Here,

the MT system is designed in all details specifically for one particular pair of languages in one direction. The source language is analyzed no more than necessary for generating output in the other language specific to the task at hand. The rules for analysis, transfer and generation are not always clearly separated.

The difference between the three approaches is illustrated by the well-known pyramid diagram shown in 1.1 (see also [Som98], 'machine translation').

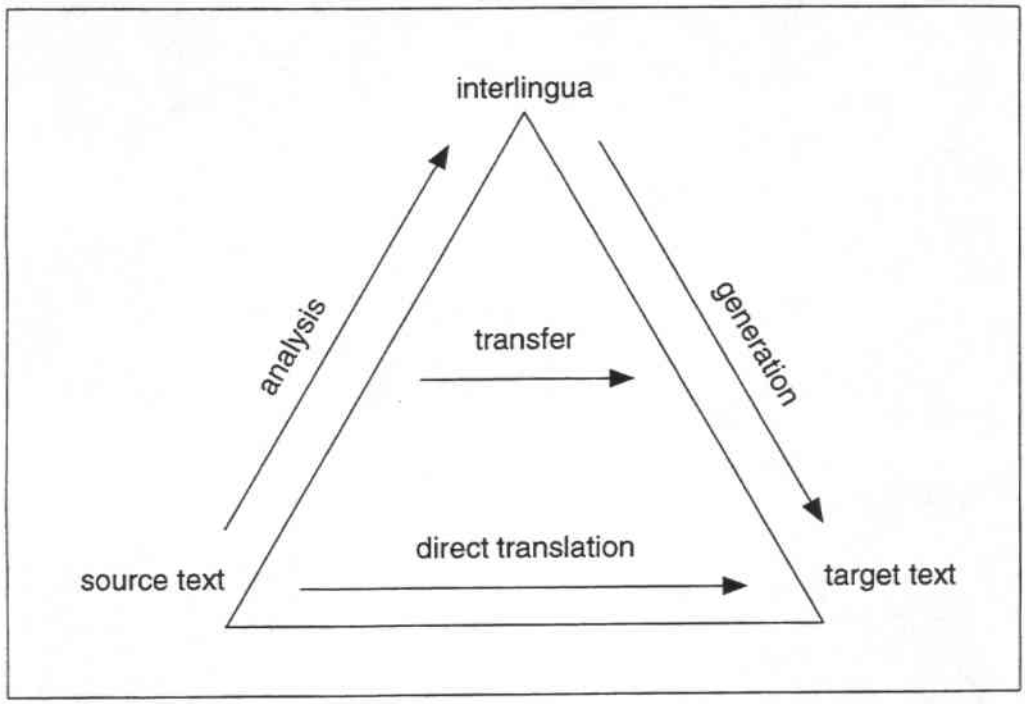

Figure 1.1: **The Pyramid Diagram:** *The deeper the analysis the less transfer is needed; the ideal case being the interlingua approach where there is no transfer at all.*

History

The *first generation direct MT* systems were essentially dictionary-based 'direct-replacement' systems. The typical translation process would involve some internal analysis of individual words (morphology), dictionary look-up

to find the target language equivalent, and then some word-order manipulations on the basis of local environment.

A turning point in the early history of MT was the infamous report of the Automatic Language Processing Advisory Committee [ALP66], which concluded that MT was slower, less accurate, and that there was no immediate or predictable prospect of useful MT. Although the report had a devastating effect on research-funding in MT, it is often said that those groups which survived, responded to it by revising the basic techniques they had been using and developing the 'indirect method'.

The fundamental idea underlying the *second generation indirect MT* systems is that the source text is transformed into the target text via an intermediate representation; whether the target text is generated directly from the representation of the source text or whether there is an intervening stage of transfer between two language-specific representations distinguishes the *interlingua approach* from the *transfer approach.*

The interlingua approach represents a theoretically purer answer to the drawbacks of the first generation approach, since the inter-lingual representation should ideally constitute an abstract representation of the meaning of the source text, capturing all and only the linguistic information necessary to generate an appropriate target text, with no undue influence from the original text. This turns out to be quite difficult to achieve in practice, however. Even the very deepest of representations that linguists have come up with are still representations of text, not of meaning, and it seems inevitable that a translation system must be based on a mechanism which transforms the linguistic structures of one language into those of the other.

However, one major problem remained, namely the question of how much 'understanding' of a text was needed to translate it. The argument is that MT must go beyond purely linguistic information, since if the machine could actually 'understand the meaning' of a sentence, it could presumably paraphrase it, answer questions about it, or translate it into another language. Some researchers have responded to this by building *knowledge-based systems* which include the ability to reason about the text they are trying to translate; e.g. the English–Japanese KBMT system developed at Carnegie Mellon University [GN91]. Although the KBMT system has been fairly successful, there are always the twin drawbacks of the human effort needed to encode all the knowledge in the first place, and then the question of scaling the system up.

In recent years, two new techniques having in common an 'empirical'

rather than 'rationalist' approach have begun to attract researchers. The *empirical approach* involves the use of corpora (i.e. large holdings of texts in machine-readable form) and statistics rather than linguistic rules.

In *example-based* systems translation is produced by comparing the input with a corpus of typical translated examples, extracting the closest matches and using them as a model for the target text. This approach is said to be more like the way humans go about translating, and is also claimed to result in more stylish, less literal translations, since it is not essentially based on structural analysis of the input.

The other main non-linguistic technique that has been proposed is the *statistics-based* approach of the IBM group [BCD+90]. It involves extracting from huge parallel corpora lexical and syntactic translation equivalents on a statistical probability basis.

A major advantage of both approaches is that, to the extent that they do not involve any linguistic theory, they do not require linguists to compile grammars or lexicons: all this is done automatically. This results in highly portable systems, since the programs developed for one language pair are entirely suitable for any other language pair, just as long as sufficient training data is available.

Nevertheless, the nowadays more successful commercial systems are almost all modelled on the 'rejected' first-generation architecture, though often with some linguistic sophistication and much computational cleverness. Thus, they still translate essentially by means of a small amount of not very exact analysis and basically word-for-word dictionary lookup.

It remains to be seen whether solid theory will eventually lead to better quality in MT than clever engineering.

1.2 Translation Quality

The practical usefulness of an MT system is determined ultimately by the quality of its output. But what counts as a 'good translation', whether produced by human or machine, is an extremely difficult concept to define precisely.

The assessment of translation quality presupposes a theory of translation. Thus different views of translation itself lead to different concepts of translation quality, and different ways of assessing it. According to [Hou98] the different approaches can be divided into a number of distinct categories: anecdotal and subjective, response-oriented approaches, and text-based approaches.

Anecdotal and subjective treatises on translation quality tend to see the quality of a translation as dependent on the translator and his personal knowledge, intuitions and artistic competence. A central problem in such treatments is the operationalization of concepts such as 'faithfulness to the original', or 'the natural flow of the translated text'. Such intuitive treatments of translation quality are atheoretical in nature, and the possibility of establishing general principles for translation quality is generally rejected.

Response-oriented approaches to evaluating translations are communicatively oriented and focus on determining the 'dynamic equivalence' [Nid64] between source and translation; i.e. the manner in which receptors of the translated text respond to it must be equivalent to the manner in which the receptors of the source text respond to the source text. Even though there is a variety of different criteria for assessing translation quality within this approach — e.g. general efficiency of the communicative process, comprehension of intent, correctness with which the message of the original is understood through the translation, or equivalence of response — all of these prove to be as vague and non-verifiable as those used by proponents of the intuitive-anecdotal approach.

Text-based approaches are based on linguistics, comparative literature or functional models. In linguistically-based approaches, pairs of source and target texts are compared with a view to discovering syntactic, semantic, stylistic and pragmatic regularities of transfer. Approaches which draw on

comparative literature assess the quality of a translation according to the literary system of the target culture. The functional theory of translation (e.g. see [RV91]) claims that it is the purpose of a translation which is all important, i.e. the communicative situation for which the translation is intended.

In the context of MT not all of these approaches are applicable. For most applications MT is not able to produce translation of 'high quality' yet — at least not fully automatic MT systems[1]. Therefore most research in MT aims at 'good-enough' translations, which basically try to achieve a content-oriented equivalence of the translation to the original.

This work concentrates on translation of spoken language in the travel-planning and scheduling domain; i.e. a client tries to make flight and hotel reservations, plan trips, or make credit-card payments with a travel agent, who speaks a different language. Here we can relax the requirement for content-oriented equivalence, since the main goal of our approach is to establish a meaningful communication between a travel agent and his/her client. The transfer of pragmatic meaning, however, can be very significant: The degree of agreement or denial can be intricately encoded in the utterance, or only given by intonation; but it is nevertheless crucial for accomplishing the task of booking a flight, making a hotel reservation or planning other travel arrangements.

1.3 Speech Translation

Since speech is our most natural form of communication, using spoken natural language to access such systems has become an important research goal. There are several specific advantages to speech as input medium. With speech as the means of access, even casual users need relatively little training before interacting with a complex system. Interactions in such a case can be quick, since speech is our fastest mode of communication, and the system user's hands are free to point, manipulate the display, and so forth. This capability is especially important in environments that place many simultaneous demands on the user.

However, for machine translation speech input constitutes a by far more

[1]It should be noted, however, that even human translations are usually subject to revision, though it should also be said, that revising MT output is quite different from revising human output.

complicated and difficult problem than written language. Spoken language, especially spontaneous speech, is hardly ever well-formed in the sense of rigid syntactic structure. It contains disfluencies, hesitations ('*um*', '*hmm*', etc.), repetitions ("... *so I, I, I guess, what* ... "), and false starts ("... *how about we meet on Tue* ... *um* ... *on Wednesday* ... "). Furthermore, phrases with low information content occur, e.g. "*let me see*", "*well*", "*i dunno*".

Yet the utterances can still be cluttered by recognition errors and environmental noises that occur during speech recording, such as coughs, laughter, telephone rings, etc. Without proper treatment, these noises may be recognized as one of the words in the vocabulary, potentially causing great damage in the translation process.

Therefore, a successful speech translation system cannot rely on perfect recognition or perfect syntax. Rather it must search for a semantically plausible interpretation of the speaker's intent while ignoring unimportant words or fragments.

A spoken dialog does not consist of sentences in the classical sense, nor are punctuation markers provided to limit them. Instead, each utterance is fragmentary and each speaker's turn often contains two or more sentences or concepts, e.g. " ... *no, Tuesday doesn't work for me* ... *how about* ... *Wednesday morning*". Even if punctuation markers were given, attempts to translate such fragmentary utterances often result in awkward output.

To provide useful spoken language communication across language barriers, we must therefore 'interpret' an utterance, or extract its main intent, rather than attempt a sentence by sentence translation.

The syntactic structure of spoken language is highly complex, and characterized by frequent sentential adjuncts. How to analyze those correctly is still an unsolved problem, but also unnecessary for most translation — usually it is sufficient to translate chunk by chunk.

1.4 Work Objective

Traditionally, parsing has been done with various symbolic approaches. They have the advantage of highly specific analyses and therefore very precise translations. The drawback of all symbolic approaches is the need for hand modelled grammars, which have to be adapted to specific languages and domains.

Especially for spoken language the major disadvantage and difficulty for the symbolic approaches lies in building the parser: it takes as input spontaneous speech, including ungrammaticalities, stops, and restarts, corrupted with speech recognition errors. The parser should output a consistent analysis in a formalism usable for processing by other components. This is usually done by modelling speech idiosyncrasies explicitly in the grammars or by 'soft' rules, that allow the parser to handle exceptions during the parse process (e.g. skipping or filling in words).

Within the last decade, statistical techniques have been proposed for learning the parsing task in order to avoid the tedious manual modelling of grammars. In this work we will focus on the parsing task in the context of a language translation system for spoken language applying the interlingua principle.

	Salt		**Soup**	
	development	porting	development	porting
development programmers	few, short-medium time	no	few, med.-long time	no
application programmers	no	one, short time	no	no
grammar writers	one, short time	one, short time	some long time	some long time
corpus annotators	many, short time	many, short time	no	no

Table 1.1: **Comparison of Efforts:** *Displayed are expected efforts involved in development and porting of a statistically learnable parser (Salt) and a conventional grammar-based system (Soup).*

We propose an alternative parsing method, which performs shallow parsing of input utterances using a probabilistic framework, that allows derivation

of model parameters automatically from a corpus, using statistical inference techniques. The expected advantages of the proposed approach lie in its economy considering time and man-power — and thus the total costs — for creating an analysis component for a speech translation system. Only for system development highly skilled programmers are required. Once the system is implemented, it can be ported to other domains or languages by (less skilled) application programmers. In both cases, development and porting, annotators can provide the training data by tagging domain specific corpora. Tagging is much less time consuming than writing grammars, and annotators can be trained in a matter of days.

Table 1.1 shows a comparison of the expected development and porting effort involved for the proposed statistical analysis system (SALT) and a grammar-based or symbolic system (SOUP). Only for development and implementation of the symbolic system skilled programmers are required. Porting is done by writing new grammars for the given parser. Grammar development is a tedious task involving skilled grammar writers for several months or even years. It is also a highly complex process, such that employing more writers in the development process cannot reduce the overall development time below a certain mark.

	Salt	**Soup**
accuracy	medium	high
economy	medium to high	medium
extensibility	high	low
input-features	variable	words only
generality	high	only parser
parse-depth	shallow	high
parse-speed	medium	medium
portability	high	medium
robustness	high	medium to high
variability	by learning	through grammar-writing

Table 1.2: **Comparison of Characteristic Features:** *Displayed are characterstic features of a statistically learnable parser (Salt) and a conventional grammar-based system (Soup).*

Besides a good portability and economy the statistical approach is also characterized by its robustness and flexibility inherent in the statistical mod-

els. They allow a simple integration of other knowledge sources into the parse process, e.g. such as prosody, lip reading, etc. Discourse models can be implemented easily within the statistical framework, and even the input can consist of other representations than word strings, e.g. speech lattices or n-best lists. Table 1.2 summarizes the characteristic features of both systems.

Chapter 2

Shallow Statistical Parsing

This chapter explains the environment in which SALT (Statistical Analyzer for Language Translation) is developed, tested and evaluated, the architecture of the overall-system, and finally the details on the machine-learning models used within our system.

2.1 System Environment

The experimental environment for SALT lies within the framework of the C-STAR II (Consortium for Speech Translation Advanced Research) machine translation effort for translating travel planning dialogues [Wai96, LGLW98, Wos, WBG+98].

2.1.1 The C-STAR II Translation Effort

The Consortium for Speech Translation Advanced Research, or C-STAR, was formed as a voluntary group of institutions committed to cooperate in the research on speech-to-speech translation. C-STAR II, a continuation of the former C-STAR group, consists of 6 partners and additional affiliate organizations from 10 countries. The partners in C-STAR II are: ATR, Japan; ISL, University of Karlsruhe, Germany; ISL, Carnegie Mellon University, USA; ETRI, Korea; IRST, Italy;and CLIPS-GETA, France. All partners commit to build a full end-to-end system each, handling their own language as input and producing at least one output language. The affiliates perform relevant basic research, participate in workshops and can optionally build

individual components in cooperation with one or more of the C-STAR II partners. Since the six main partners are responsible for one input language each, the C-STAR-II setup allows for a minimum of six languages: German, French, English, Japanese, Korean and Italian.

Within this framework, a translation system was developed for the broad domain of travel planning, which contains a rich structure of sub-domains, such as hotel reservation, transportation, sightseeing, and scheduling.

2.1.2 The C-STAR II Interchange Format

This section describes an interlingua for machine translation of spoken travel planning dialogues. The interlingua used by C-STAR is known as interchange format (IF). The C-STAR languages are English, French, German, Italian, Japanese, and Korean. C-STAR adopted an interlingua in order to facilitate translation between as many language pairs as possible with minimal effort. Sites that wish to use IF supply an analyzer that produces IF from sentences in the home language and a generator that takes IF as input and produces sentences in the home language. Using the analyzer from one language and the generator from another results in translation from the first to the second language.

Advantages

There are four main advantages for an interlingua approach in this domain:

First, it helps to reduce the dependence of the output sentence on the structural form of the input language. What matters is the meaning or intent of the input utterance, however it was expressed by the speaker: Sentences like "*I don't have time on Tuesday*", "*Tuesday is bad*", or "*I'm on vacation, Tuesday*", can all be mapped onto the same intended meaning "I am unavailable on Tuesday", and an appropriate sentence in the output language can be generated. Even culturally dependent expressions can be translated in a culturally appropriate fashion. E.g. the closing phrase "*Thank you for using World Wide Travel*", which does not sound appropriate when translated directly into German, could be translated into "*Vielen Dank für Ihren Anruf.*" literally: "*Thanks for your call.*".

The second advantage of the interlingua approach, which we have already mentioned above, is the comparative ease by which additional languages can be added. In the current system, only the language specific grammars for the

analyzer and generator modules have to be developed in order to integrate a new language into the system.

The ease of generating output in any one of the supported languages, given the analysis of the original input as interlingua representation constitutes the third advantage: it allows generation of a paraphrase in the source language by which the user can verify if an input utterance was properly analyzed. This feed-back feature strongly helps to improve the usability of the system.

The fourth and last advantage of the interlingua approach within the C-STAR project lies in the ability to exchange already analyzed utterances (i.e. the 'meaning' of the utterances within the given domain) between the sites of the C-STAR partners. Since many of the partners chose to implement translation systems based on an interlingua approach using the same canonical interlingua representation, i.e. the C-STAR interchange format (IF), analysis and generation can be split across the partner sites. Thus if one site provides an IF representation for a given utterance, all other sites supporting the C-STAR IF can generate a translation into their target languages without further analysis of the original utterance.

Design

The most important factor in the design of an interlingua is that it must abstract away from peculiarities of any particular language in order to allow for translations that are non-literal, but capture the speaker's intention or *speech act*, such as giving or requesting information, or introducing oneself, etc. Taking the notion of speech acts one step further, one can identify *domain actions* such as requesting information about the availability of a hotel, or giving information about the price of a hotel room. Apart from the speaker's intention domain actions also capture the relevant topics or *concepts* of an utterance, such as availability, reservation, hotel or room. More specific information, e.g. prices or times, is represented by the *arguments* of a domain action.

However, the C-STAR project represents a special challenge to an interlingua approach, since it requires an interlingua to be used at multiple research sites. It was therefore necessary to design a simple interlingua that could be used reliably by many MT developers. Simplicity is possible largely because the dialogues in the C-STAR project deal with travel planning, which is a task-oriented domain with clearly identifiable domain actions. These domain

actions form the basis of the C-STAR interlingua.

Thus, the IF does not represent literal meaning of an utterance and is far-removed from the source language syntax. It represents only the domain actions that the utterance is intended to perform.

Therefore the three main design principles for the interchange format are:

- The interchange format is based on domain actions.
- It is compositional, i.e. domain actions are built from an inventory of sub-categories (speech acts, concepts, and arguments).
- Its representation should be suitable for all C-STAR languages.

Implementation

Domain Actions (DA) consist of three representational levels: the speech act, the concepts, and the arguments. In addition, each DA is preceded by a speaker tag, i.e. the side information which indicates who is speaking ('`a:`' for the agent, '`c:`' for the client). The speech act and side information are obligatory, whereas the concepts and the arguments are optional. Plus signs, '+', separate speech acts from concepts and concepts from each other. The general scheme of a DA can be roughly characterized as shown in (2.1). However, there are constraints on the order of the concepts so that not all combinations are possible.

(2.1) `speaker: speech act +concept* (argument*)`

In example (2.2) the speech act is `give-information`, the concepts are `availability` and `room`, and the arguments are `time` and `room-type`. The possible arguments of a DA are determined by inheritance through a hierarchy of speech acts and concepts. In this case `time` is an argument of `availability` and `room-type` is an argument of `room`. Example (2.3) shows a DA which consists of a speech act with no concepts attached to it. the argument `time` is inherited from the speech act `closing`. Finally, example (2.4) demonstrates a case of a DA which contains neither concepts nor arguments.

(2.2) On the twelfth we have a single and a double available.
```
a:give-information+availability+room
(room-type=(single & double),time=(md12))
```

(2.3) And we'll see you on February twelfth.
`a:closing (time=(february, md12))`

(2.4) Thank you very much
`c:thank`

The DAs in the given examples do not capture all of the information present in their corresponding utterances. For instance they do not represent comparatives, relative clauses, extensive noun modification, modality, politeness, formality, certainty, tense, etc. Some of this information will need to be added in the future. For example, embedded clauses or noun modifications can contain relevant information for the current communicative situation. Other features like tense are often (but not always) predictable from the DA. And others like modality (e.g., the '*could*' in "*Could you tell me ...* ") are generally part of the formulaic, conventional ways of expressing the DAs in specific languages, but their form is not relevant for translation. As mentioned in section 1.2 the representation of some of these features is not really needed for translating utterances in a given task-oriented context, since they only indirectly contribute to the identification of the DAs.

Example (2.5) shows the English paraphrase and the German translation for sentence (2.2).

(2.5)

Input:	On the twelfth we have a single and a double available.
Paraphrase:	A single and a double room will be available the twelfth.
German:	Es gibt Einzelzimmer und Doppelzimmer am zwölften.

The following paragraphs describe the four components of DAs, speaker tags, speech acts, concepts, and arguments in more detail:

Speaker Tag: The speaker tag is either '`a:`' for agent or '`c:`' for client to indicate who is speaking. The speaker tag is sometimes the only difference between the IFs of two different sentences. For example, "*Do you take credit cards?*" (uttered by the client) and "*Will you be paying with a credit card*" (uttered by the agent) are both requests for information about credit cards as a form of payment.

Speech Act (SA): There are currently 40 speech acts defined in the IF. Some speech acts are very general. For example, `give-information` is used

in many DAs where the speaker's intent is to inform the listener of something, such as `give-information+temporal+departure+flight`, `give-information+expiration-date`, etc. Others are more specific, such as `delay-action`, which is used specifically for utterances like "*I'll get back to you on that*". Normally each DA has one speech act. However, there are three special speech acts that combine with other speech acts. These are `verify`, `request-verification`, and `negate`. For example the sentence "*So you're not leaving on Friday, right?*" has the speech act `request-verification-negate-give-information`.

Concepts: There are currently 68 concepts defined in the IF. Each DA can have zero or more concepts following the speech act, although not all possible strings of concepts are allowed. Concepts fall into several classes that roughly constrain how they combine with each other. Some classes of concepts are actions (change, reservation, confirmation, cancellation, etc.), attributes (availability, size, temporal, price, location, features, etc.), and entities (room, hotel, flight, numeral, expiration date, etc.). The usual order of concepts in a DA is `action+attribute+entity` as in `request-action+reservation+temporal+room` for "*I'd like to make a reservation for a room on the fifth*". In this case, the speech act is `request-action` and the concepts are `reservation`, `temporal`, and `room`.

The concept components of a DA capture the focus of a sentence. For example, the sentence "*The week of the twelfth we have both singles and doubles available*" mentions a date, a room type, and the notion of availability. However, since the focus of the sentence is availability, the dialogue act is `a:give-information+availability+room` and the time and room type are expressed as arguments of this dialogue act.

Arguments: Arguments add specific information to the DA, such as times, prices, and specific features of entities. An argument consists of an argument name and a value separated by an equal sign, for example `room-type=double`. In addition to atomic values, there are various types of complex values as shown in examples (2.6)-(2.14). Multiple values and coordination can combine with price, time, interval, frequency, and duration for arguments like "*on July 5 and July 6 at 4:00*".

(2.6) **multiple values:**
`room-type=(double,non-smoking)`

a non-smoking double

(2.7) **coordination:**
`room-type=(single & double)`
a single and a double

(2.8) **quantity:**
`room-type=(double, quantity=2)`
two doubles

(2.9) **price:**
`price=(currency=dollar, quantity=50, per-unit=night)`
fifty dollars per night

(2.10) **time:**
`time=(md5, tuesday, july, 1998, 16:00, afternoon)`
Tuesday July 5, 1998 at 4:00 in the afternoon

(2.11) **time interval:**
`time=(start-time=(md5, july), end-time=md10)`
from July 5 to 10

(2.12) **duration:**
`duration=(time-unit=day, quantity=9)`
for nine days

(2.13) **frequency:**
`frequency=(time-unit=hour, quantity=2)`
every two hours
`frequency=(per-unit=hour, quantity=2)`
two times per hour

(2.14) **lists of characters:**
`spelling=[g, a, t, e, s]`
g a t e s

The possible arguments of a DA are determined by the speech acts and concepts it contains. For example, `give-information+temporal+flight` can take the arguments associated with the concepts `temporal` (time, duration, frequency) and `flight` (flight-type, carrier-name, flight-number, destination, origin). There are currently 86 argument names defined in the IF.

2.1.3 The Janus-III Translation System

The Janus project at the Interactive Systems Laboratories[1] constitutes an ongoing effort to develop a machine translation system specifically suited for spoken dialogue [WJM+91, WJM+92, OAM+92, WAWB+94, GSB+95, Wai96, LGLW98, WBG+98]. It was one of the early systems designed for speech translation, developed at Carnegie Mellon University and University of Karlsruhe in the late '80s and early '90s. Since then it has been extended at both sites to more advanced tasks.

While the first version, Janus-I, processed only syntactically well-formed (read) speech over a small (ca. 500 words) vocabulary, the most recent version of Janus, the Janus-III Translation System, now copes with spontaneous conversational human-human dialogs in limited domains with much larger (several thousand words) vocabularies. At present it accepts English, German or Spanish input and produces translations into English, German, Japanese, Korean, and Spanish.

System Design

Janus-III was designed to accommodate multi-party, multi-lingual conversations between travellers and travel agents. It's design is based on the following four main principles: an interlingua-based approach, semantic grammars, modular grammars, and an efficient integration of multiple grammars.

The system exemplifies an *interlingua-based* approach (see section 1.1) and consists basically of three modules: speech recognition, analysis, and generation. The analyzer and generator are language-independent, i.e. they consist of a general processor that can be loaded with language specific knowledge sources. This allows for easy expansion of the system to new languages. Since each language is usually integrated as both a source and a target language, input analyzed into an interlingua representation can then be translated back into the source language (in our case, English), which results in a paraphrase of the input. The paraphrase can be used as a mechanism for verifying analysis and representation correctness, as well as for end-to-end evaluation purposes. The interlingua representation can also be exported to the generation systems of other C-STAR II partners for translation into languages not supported at the Interactive Systems Labs.

[1] The Interactive Systems Laboratories are jointly located at Carnegie Mellon University in Pittsburgh and at University of Karlsruhe in Germany.

Semantic grammars are used for both analysis and generation. Analysis using semantic grammars has been shown effective in providing accurate translations for limited domains; and it is also known to be more robust against ungrammaticalities in spontaneous speech and recognition errors in speech-to-speech translation systems [LLZ+97, MGS+95]. Rather than focussing on the syntactic structure of the input, semantic grammars directly describe how surface expressions reflect the underlying semantic concepts that are being conveyed by the speaker — their non-terminal nodes represent semantic concepts and not syntactic categories. Because they focus on identifying a set of predefined concepts, they are relatively well suited for handling the types of meaningful but ungrammatical disfluencies that are typical of spontaneously spoken language, and are also less sensitive to speech recognition errors. Semantic grammars can also be developed relatively fast in a limited domain, where the set of relevant concepts is relatively small; but expanding them to cover new domains is usually quite hard and complicated. Each new semantic concept requires new rules to be added to the existing grammars, since syntactic generalities usually cannot be utilized. For large domains, this results in very cumbersome grammars that are intricate to expand and develop, and which become highly ambiguous in nature.

In Janus-III, the development of *modular grammars* and common libraries copes with the problems of expanding semantic grammars to new domains. Modularization and the use of shared common libraries are a well-established concept in software development. Many of the advantages of modularity and shared libraries equally apply to the design of semantic grammars in large domains, particularly if the domain can be dissected into multiple subdomains. In the current Janus system, grammars are separated into the subdomains for hotel reservations, transportation, sightseeing, and scheduling; each sub-grammar covers the specific set of semantic concepts related to its sub-domain. An additional grammar provides cross-domain concepts, such as common openings and closings. The sub-grammars also draw from a shared library of rules in order to maintain consistency in the analysis of time and date expressions, name phrases, auxiliary verbs, etc. The shared library and the cross-domain sub-grammar substantially reduce the effort in expanding the system to new domains.

The *integration of multiple grammars* is done in a common analysis module. It consists of the robust SOUP parser [Gav98, GW98] which is able to analyze the input using multiple grammars concurrently. Segmentation of long utterances is performed as part of the parsing process. Thus, the parser

analyzes complete utterances using all sub-domain grammars, and produces a lattice of parse trees that contains all possible ways of segmenting the utterance according to the different domain grammars. The analyzed segments are tagged with a domain-tag that reflects the sub-grammar that was used in creating the analysis. The lattice of all possible parsable segments is then used in a statistical domain re-scoring procedure, in order to resolve ambiguities that arise from the combination of multiple domain grammars. The search for the optimal sequence of parsed segments is performed within the SOUP parser at the end of the analysis stage. The parser then outputs a ranked list of possible sequences of parse trees for the entire utterance.

System Architecture

Figure 2.1 shows a component diagram of the complete Janus speech translation system for the travel domain. The main system modules are speech recognition, analysis, and generation.

The interface between the speech recognition module and the translation system is via an N-best list of text string hypotheses in the source language. Translation is then performed by analyzing the text string in the source language into an interlingua representation, and then generating a string in the target language. Analysis is done by SOUP, a robust parser designed for spoken language analysis. Since SOUP works with semantic grammars, the parser analysis contains all necessary semantic information. Therefore only a simple format conversion, not contributing any significant information beyond that derived by the parser, has to be done in order to yield an interlingua representation. This is done by the Parser-to-IF mapper, which converts the output-representation of the parser into a canonical interchange format (IF), which we have described earlier in section 2.1.2. The IF interlingua representation is then passed on to generation and speech synthesis.

2.2 System Model

Statistical methods in machine translation rely mainly on information that can be extracted automatically from corpora of collected natural language. A small community has experimented with either purely statistical approaches [BCD+90, Sch93, Rie94], or connectionist based approaches [Ber91, MD91, Jai91, BPW94, WW94, Buø96, BW96a, BW96c, BW96b]. Their main ad-

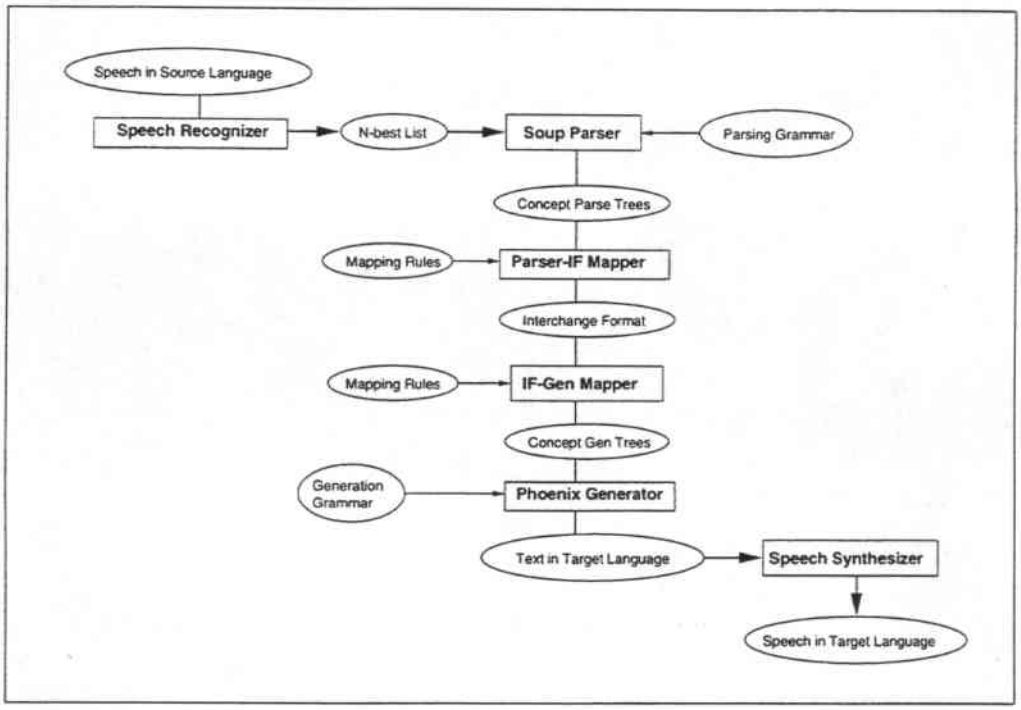

Figure 2.1: **The Janus-III Machine Translation System:** *Component diagram of the Janus speech translation system for the travel domain reproduced from [LGLW98].*

vantages are learn-ability and robustness. It is also believed, that — for a limited domain — translation systems based on statistical methods can be implemented in a fraction of the time that the development and implementation of a system based on a symbolic or linguistic approach would need.

However, many of the above cited approaches have suffered from a number of disadvantages: For example, some approaches contain no or only limited linguistic information to be used in translation or understanding. For others, no clear and quantitative statement about the overall performance is made, or it has not been evaluated with real world data. But the most pertinent problem of most approaches is that they require millions of training sentences.

In the following section we will motivate our approach to shallow statistical parsing for language translation within the framework of the Janus-III system. Furthermore, we will discuss the important design issues, and give a formal specification of the computational model underlying our approach.

2.2.1 Motivation

A weakness of many grammar-based analysis systems is that they are not very robust to concept phrasings that deviate from those expected in the grammars, or to the occurrence of unexpected 'noise' within concepts, which can easily occur in a system based on speech input (see section 1.3).

Another significant drawback of all symbolic approaches is the large amount of work involved in the design and implementation of hand modeled grammars, which also have to be adapted to each language and domain specifically. Further, for achieving a certain level of robustness, which is needed in an environment dealing with speech translation, extra hand modelling effort is required.

To address these problems is one of the main concerns of this work. It describes an alternative parsing method that combines both statistical and grammar information. Statistical information will be used in order to segment and label utterances at the DA level. I.e. each utterance gets split into segments corresponding to its DAs, and the segments are labelled with the most likely speech act. Each segment is then taken further apart into 'argument chunks' which roughly correspond to those strings of words in the utterance, that fill in the argument values. In a second step the concepts of each DA are determined, with respect to the already recognized speech act and arguments. A modified version of the grammars for parsing just argument fragments is then used in a last step in order to extract the appropriate values of the detected argument fragments from the utterance.

Statistical identification of DAs and a number of related problems that make use of a 'chunk and label' paradigm have been studied by various authors in the recent past [NM94, WKNN97, REKK96, RK97, TKI+97, JBC+97, FLL+98, SSB+98, Rie99]. Preliminary experiments on statistical DA extraction in the Janus project have shown encouraging results [FKWT98].

2.2.2 Design

The integration of a new analysis module within the framework of the Janus-III machine translation system requires replacement of the SOUP parser plus, optionally, one or more of the consecutive components in the translation process. This means essentially building a system that works with recognized utterances as input and produces output that can be fed back into the system at one of the succeeding stages, such as the Parser-to-IF mapper, the IF-to-

Generation mapper, or the Phoenix Generator.

For our approach we chose the simplest and most feasible way of integration by producing output that is similar to SOUP output, and thus can be fed directly into the Parser-to-IF mapper. There are several reasons and advantages of this design decision:

First, we have two well-defined and already implemented interfaces to the Janus system, namely the N-best list of text string hypotheses from the recognizer as input interface, and the concept parse trees required by the Parser-to-IF mapper as output specification. Therefore, no other modules are affected or need to be changed in order to integrate our system.

Since we use supervised machine learning approaches, such as hidden understanding models, language models and neural networks, the second advantage lies in the strictly specified input-output relationship that these already existing interfaces impose on the design of our system: Using the SOUP parser we can generate training data for the supervised training of our system basically for 'free'. The original utterances fed into SOUP for generating a database of training examples have been collected within the C-STAR project. They were created either for testing newly added grammar-rules, or they stem from user studies evaluating the C-STAR II user interface.

Another advantage of the chosen integration lies in the simplicity of combining the output of our analysis module with the output of the SOUP parser in order to improve the over-all system performance. Since both modules (SALT and SOUP) provide the same output representation a comparison and evaluation of their output is straightforward.

However, it should be mentioned here that the actual combination of both modules is done at the IF-level, i.e. after the Parser-to-IF mapper has transformed the parse tree representation into valid IF format. We chose an integration at this stage, since the statistical models used to segment and label utterances, tend to over-generate — i.e. they predict more arguments and DAs than are actually present in the given utterance. Producing too many segments or segments that are too small, especially at the argument level, can result in invalid IF. Therefore, we implemented a post-processing module, that filters the output at IF level and eliminates all invalid and inconsistent parts of the IF representation of an utterance (see section 2.3.5 for examples).

Figure 2.2 shows a component diagram of the resulting multi-engine translation system. Multi-engine translation was proposed by Frederking et al. [FNF+94] and has since been implemented in the Diplomat [FRH97] and

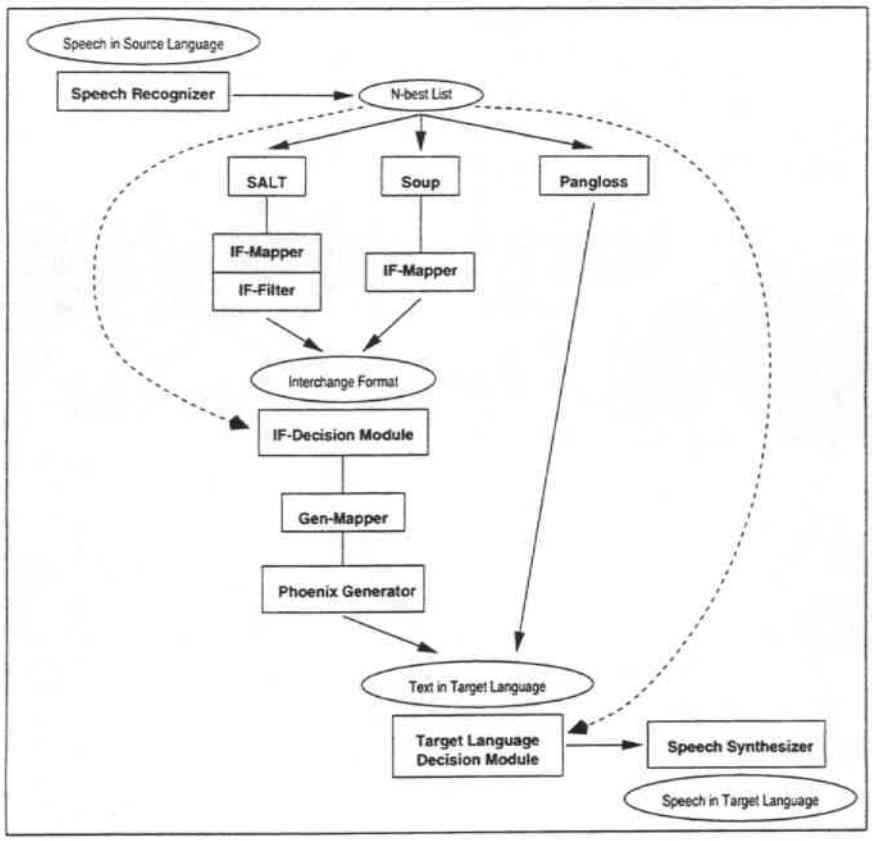

Figure 2.2: **The Multi-Engine Machine Translation System:** *Integration of Soup, Salt and Pangloss within the framework of the Janus-III speech translation system.*

Verbmobil [Wor98] systems. A multi-engine system applies multiple translation programs simultaneously and makes a translation by composing the best parts from the various outputs. Typically, a multi-engine system might include knowledge-based, statistical, and direct dictionary based approaches. In our case the components are SALT, the statistical analysis module described in this work, SOUP, the grammar-based analysis module of Janus-III, and Pangloss, a direct-translation system based on a collection of examples, glossary and dictionary lookup [FNF+94].

The decision modules shown in the diagram can be basically thought of as engine dependent language models, that try to predict which engine will most likely come up with the best translation given the original utterance. In cases

where the best predicted translation contains obvious errors or inconsistencies a fall-back mechanism has to be applied, and the next best translation from one of the other engines has to be chosen.

2.2.3 Specification

Our goal is to perform shallow parsing of input utterances using a probabilistic framework, which enables us to derive model parameters automatically from a corpus, using statistical inference techniques.

Given the sequence of words $W = w_1, w_2, \ldots, w_N$ of an utterance , we attempt to extract the sequence of domain actions $D = d_1, d_2, \ldots, d_K$ that has the highest posterior probability $P(D|W)$ given the input word sequence:

$$D^* = \underset{D}{\operatorname{argmax}}\, P(D|W) \tag{2.15}$$

Further, each domain action d_k consists of a speech act combination s_k, an optional combination of concepts c_k, and a sequence of argument segments A_k:

$$d_k = (s_k, c_k, \underbrace{a_{m_{k-1}+1} a_{m_{k-1}+2} \ldots a_{m_k}}_{=A_k}) \quad \text{for} \quad k = 1, \ldots, K \tag{2.16}$$

Each argument segment a_m consists of a label and a sequence of words W_{a_m}:

$$W_{a_m} = w_{n_{m-1}+1} w_{n_{m-1}+2} \ldots w_{n_m} \quad \text{for} \quad m = 1, \ldots, M \tag{2.17}$$

Consequently, each speech act s_k consists of its label and the sequence of words W_{s_k} that covers all words of its accompanying arguments:

$$W_{s_k} = (W_{a_m})_{a_m \in A_k} \quad \text{for} \quad k = 1, \ldots, K \tag{2.18}$$

Thus, an utterance gets segmented at two levels: first, according to (2.17), the words $w_1, w_2, \ldots, w_N$ are grouped into segments $W_{a_1}, W_{a_2}, \ldots, W_{a_M}$ with respect to the argument segments $a_1, a_2, \ldots, a_M$. Then, in accordance with

(2.16), the argument segments are grouped into sequences $A_1, A_2, \ldots, A_K = A$ and combined with speech act labels $s_1, s_2, \ldots, s_K = S$ and concept labels $c_1, c_2, \ldots, c_K = C$ to domain actions $d_1, d_2, \ldots, d_K = D$. This two-level segmentation is illustrated by an example utterance given in figure 2.3.

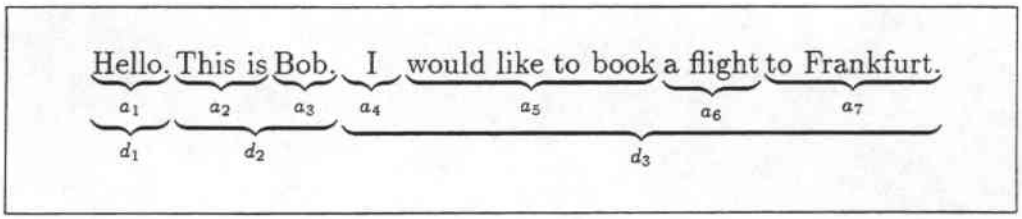

Figure 2.3: **The Segmentation of an Example Utterance:** *Two-level segmentation of an example utterance into argument segments $a_1, \ldots, a_7$ and domain actions $d_1, \ldots d_3$.*

Given this decomposition of D, we can re-write $P(D|W)$ as:

$$
\begin{aligned}
& P(D|W) \\
& \approx P(S, C, A|W) && (2.19) \\
& = P(C|S, A, W) \cdot P(S|A, W) \cdot P(A|W) \\
& \approx P(C|S, A, W) \cdot P(S|A) \cdot P(A|W) && (2.20) \\
& = P(C|S, A, W) \cdot \frac{P(S) \cdot P(A|S)}{P(A)} \cdot \frac{P(A) \cdot P(W|A)}{P(W)} \\
& = \tfrac{P(S)}{P(W)} P(C|S, A, W) \cdot P(A|S) \cdot P(W|A) \\
& \approx \tfrac{P(S)}{P(W)} \prod_{k=1}^{K} P(c_k|s_k, A_k, W_{s_k}) \prod_{k=1}^{K} P(A_k|s_k) \prod_{m=1}^{M} P(W_{a_m}|a_m) && (2.21)
\end{aligned}
$$

If we replace $P(D|W)$ in equation (2.15) by (2.21) we get equation (2.22), which describes the computational model for estimating $\tilde{D}^*$, the proposed approximation to D^*:

$$
\tilde{D}^* = \underset{D=(C,S,A)}{\operatorname{argmax}} P(S) \prod_{k=1}^{K} P(c_k|s_k, A_k, W_{s_k}) \prod_{k=1}^{K} P(A_k|s_k) \prod_{m=1}^{M} P(W_{a_m}|a_m) \tag{2.22}
$$

In the derivation of equation (2.22) we make the following approximations:

First approximation (2.19): We assume that $P(S, C, A|W)$ models $P(D|W)$ reasonably well without integrating a model for argument substructures within our statistical framework. This assumption is supported by experimental evidence stating good results for segmentation and labelling at argument level without a sub-argument model.

Second approximation (2.20): Here we assume that the speech acts S only depend on the arguments A and not on the words W. As we know from the specification of the interchange format in section 2.1.2, this assumption is not true in general, since the specification allows for domain actions without any specified arguments. Also the semantic grammars used within the SOUP parser detect speech acts based on words, and not arguments. But we can circumvent this dependency by introducing 'pseudo-arguments' that capture all speech act relevant word information at argument level. Figure 2.4 shows an example, where not all words of a given utterance are covered by arguments of the interchange format. Here we have to introduce pseudo-arguments for a_1, a_2, and a_5 in order to capture the remaining words of the utterance under an argument segment. We will describe this approach in more detail in section 2.3.2. However, it should be mentioned here that experiments show that this approach allows reliable prediction of speech acts based on recognized arguments and pseudo-arguments only.

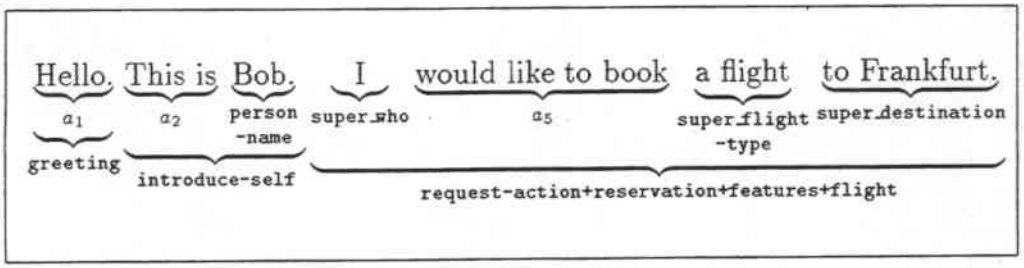

Figure 2.4: **The Labelling of an Example Utterance:** *Since the interchange format does not provide argument labels for all words within an utterance, we have to introduce 'pseudo-arguments' for a_1, a_2, and a_5 here. Note, that only the third domain action in this example contains a concept label.*

Third approximation (2.21): In equation (2.21) we make several independence assumptions at once:

First, in approximating $P(C|S,A,W)$ by $\prod_{k=1}^{K} P(c_k|s_k,A_k,W_{s_k})$ we assume that the c_k are independent from each other, and that each c_k only depends on speech acts, arguments or words within its domain action segment. Since our model — as of now — works only at the utterance level, this assumption appears to be reasonable. If we want to extend our model to the

dialogue level, we have to revoke this independence assumption, since people in a dialogue tend to re-use concepts, arguments or words, or pick them up from their conversational partners.

Second, by replacing $P(A|S)$ with $\prod_{k=1}^{K} P(A_k|s_k)$ we assume that the sequence of arguments A^k of each speech act s_k is independent from other arguments, and that A^k is independent from all speech acts other than s_k. This assumption mirrors design issues of the interchange format — namely that DAs are fully self-contained, i.e. each DA and its sub-elements are fully independent from other DAs.

Third, by substituting $P(W|A)$ with $\prod_{m=1}^{M} P(W_{a_m}|a_m)$ we assume that the words in an argument fragment do not depend on other arguments or words. Again, this assumption appears to be reasonable at an utterance level, but if we look at a whole dialogue we might have to re-consider this assumption, too.

Markov Modelling

To make the model tractable, we would like to model the probabilities given in equation (2.22) with n-gram models. We can do this reasonably well for $P(S)$, if we assume that the prior distribution of S is Markovian. This means, that each s_k depends only on a fixed number n of preceding speech acts — n is the order of the Markov process describing S:

$$P(s_k|s_1, \ldots, s_{k-1}) = P(s_k|s_{k-n}, \ldots, s_{k-1}) \tag{2.23}$$

The n-gram based discourse grammars we use for modelling the speech act sequence have this property. As experiments in section 3.1 show, $n = 1$ is a very good choice, i.e., conditioning on speech acts more than one removed from the current one does not improve the quality of the model significantly.

Further, each $P(A_k|s_k)$ for $k = 1, \ldots, K$ can be modelled in a similar way by a speech act dependent n-gram model $P_{s_k}(A_k)$, if we assume that the distribution of A_k is Markovian given the underlying speech act s_k is known.

Similarly, each $P(W_{a_m}|a_m)$ for $m = 1, \ldots, M$ can be approximated by an argument dependent n-gram model $P_{a_m}(W_{a_m})$, if the distribution of the words in W_{a_m} is assumed to be Markovian, given the underlying argument a_m is known.

The importance of the Markov assumption for the discourse grammars is that we can now view the whole system of discourse grammars and local

likelihoods as a multi-level hidden Markov model[2] (HMM). Figure 2.5 shows a structural diagram of the resulting multi-level hidden Markov model used for speech act and argument detection. It should be noted, that each level constitutes a fully connected or 'ergodic' hidden Markov model, since the given discourse models allow state-transitions, where every state of the model can be reached (in a single step) from every other state of the model. Further details of this model will be discussed in section 2.3.2.

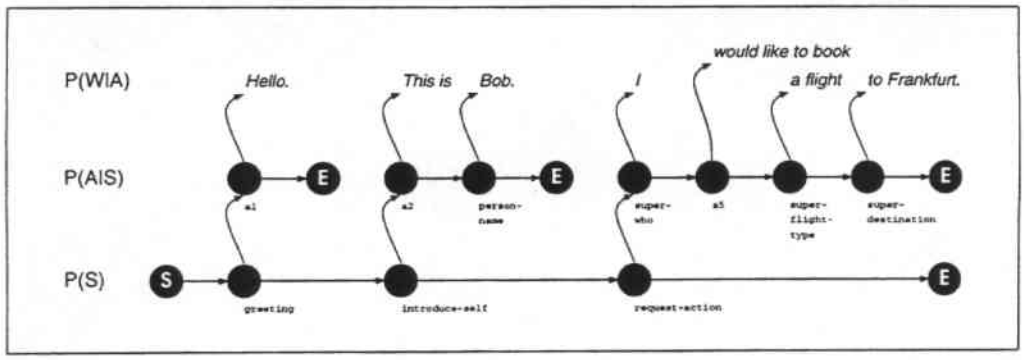

Figure 2.5: **The Multi-Level Hidden Markov Model:** *HMMs for discourse modelling at speech act and argument level are combined in a two-level structure. The output distributions $P(W|A)$ and $P(A|S)$ are distributions over word and argument sequences, respectively.*

Connectionist Modelling

Nevertheless, for estimating the term $P(c_k|s_k, A_k, W_{s_k})$ of equation (2.22) we cannot use n-gram models, since they are generative in nature and cannot deal with multiple dependencies. As described in section 2.3.3, we use an artificial neural network in order to recognize concept combinations in this approach.

2.3 System Architecture

Figure 2.6 shows a component diagram of the SALT system for shallow statistical analysis and grammar-based argument parsing for speech translation

[2]For references on hidden Markov Models see [RJ86, Rab89, SSB+98, Rie99, SRC+99]

within the framework of the Janus-III system. The complete process of parsing is divided into four stages, namely pre-processing, statistical analysis, argument parsing, and IF-post-processing.

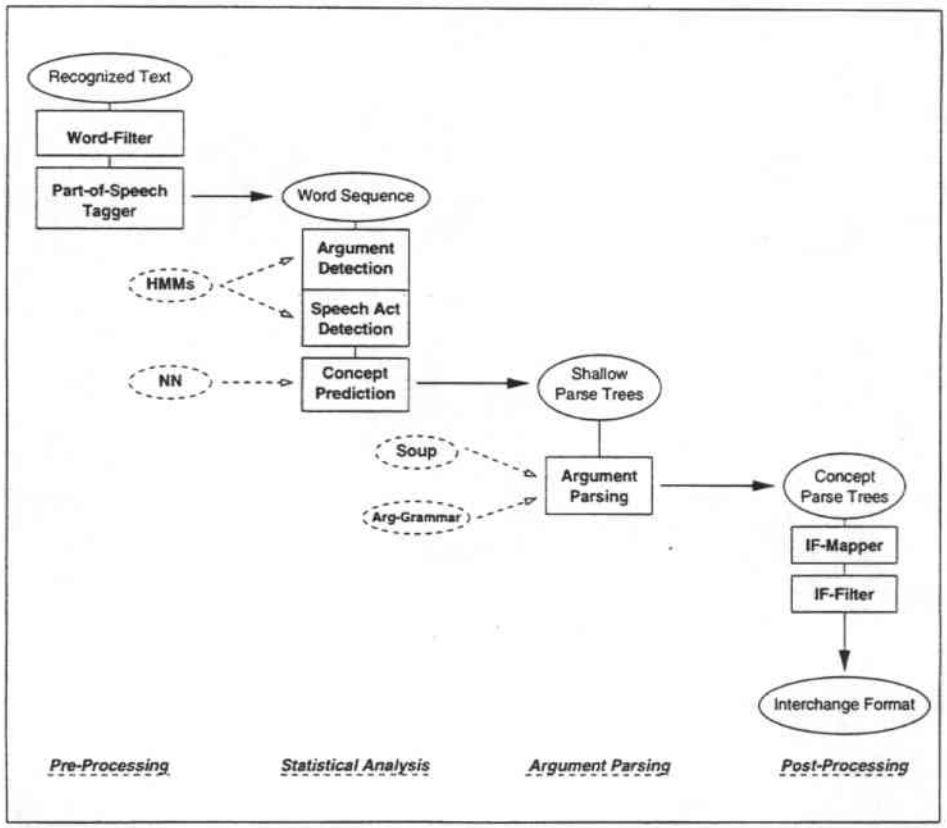

Figure 2.6: **The Statistical Analyzer for Language Translation (SALT):** *Component diagram of the proposed shallow statistical parsing approach. The part-of-speech tagger, the Soup parser, and the IF-mapper are integrated external components.*

The interface between SALT and the Janus speech recognition module is via text string hypotheses of utterances in the source language. The parse process then proceeds as follows:

First, a pre-processing is carried out by applying a word-level filter and a part-of-speech tagger to the recognized utterance (section 2.3.1).

Then, a statistical analysis is performed, which constitutes the core of the parse process. It involves multi-level hidden Markov models (HMMs)

that segment and label word sequences at argument and speech act level (section 2.3.2), followed by a connectionist concept prediction (section 2.3.3). The output representation of the statistical analysis consists of shallow parse trees — shallow in the sense that the nested substructure of the argument segments is missing.

The argument substructures get filled in at the next stage, the grammar-based argument parsing (2.3.4), resulting in a representation of concept parse trees. This requires running the SOUP parser on each detected argument segment with the appropriate argument-grammar[3].

Finally, in the post-processing stage, the concept parse trees are mapped to interchange format by the IF-mapper, which is then checked by an IF-filter in order to eliminate format errors or inconsistencies (2.3.5).

2.3.1 Pre-Processing

The pre-processing stage of SALT consists of two components: a word-level filter and a part-of-speech tagger.

Word-Level Filter

The word level filter carries out a string replacement on the recognized input word string — i.e. certain words or combinations of words in the input string are replaced by specified substitutes, e.g. '*let+s*' gets replaced by '*let us*'. The string replacement serves several purposes:

First, it helps the system to deal with recognizer output that contains abbreviations or short forms of words such as '*+bout*' for '*about*', '*+kay*' for '*okay*', or '*+til*' for '*until*'.

Further, it expands colloquial abbreviations of word combinations to their full forms, e.g. '*i+m*' gets expanded to '*i am*', '*won+t*' to '*will not*', or '*can+t*' to '*can not*'. Note, that the recognizer uses a plus sign "+" instead of an apostrophe "'" and does not distinguish between upper-case or lower-case letters.

It should be mentioned here that the expansion of these abbreviations and short forms to their full form guarantees, that the succeeding stages of the system will always get the same 'surface form' — here: their full form —

[3]In fact, as explained in the referred section, we use the whole set of the C-STAR domain grammars rather than several smaller argument grammars for parsing the argument segments.

as input. For example, no matter if a speaker would say '*can't*', '*cannot*' or '*can _pause_ not*' we will always get '*can not*' after filtering.

This a very prominent point for the succeeding statistical analysis, since it helps to reduce complexity in the input representation. Especially n-gram models usually suffer severely from inconsistent or, due to that, sparse training data. The so-called 'sparse data problem' will be addressed in the next section.

The idea of consistent surface forms is taken one step further by mapping similar phrases to more consistent representations. For example, the mapping of '*once*' to '*one time*', '*twice*' to '*two times*', and '*thrice*' to '*three times*' falls in this category; as well as the mapping of '*twelve o+clock noon*', '*twelve noon*', and '*noon*' to '*twelve pm*'.

Some colloquial phrases get changed to their more proper forms, e.g. '*a couple a*' is mapped to '*a couple of*', while others are completely taken out, e.g. '*by the way*', '*by any chance*', '*actually*', '*basically*', or '*in fact*'. This can be done here, since in the given the task of travel-planning and scheduling those phrases carry no meaning for the actual conversation.

The complete list of string mappings and replacements, as it is processed in the word-level filter, is given in table A.1. The overall effect of the filter is, as already mentioned, a reduction of complexity in the input to the succeeding stages of our system. This is achieved by mapping strings of words which represent the same meaning, or sometimes only the same structure, given task at hand, onto consistent surface forms.

The most important reason for applying this specific word-level filter in the pre-processing stage of our system is the need to stay consistent with the given training data. All the training data for SALT is generated with help of the SOUP parser, which carries out the same string mappings and replacements before parsing as is implemented in our word-level filter.

Part-of-Speech Tagger

According to Zipf's law [Zip29] the relative frequency of the i'th most frequent word in a given corpus is proportional to $1/i$. Figure 2.7(a) shows, that the relative word frequencies in our training corpus follow approximately a Zipf-like distribution — i.e. they are proportional to α/i^{β} with $\alpha \approx 0.2$ and $\beta = 2/3$.

Most words occur highly frequent due to peculiarities of the underlying language; e.g. '*a*', '*is*', '*the*', '*that*, or '*to*' are frequently used words in written

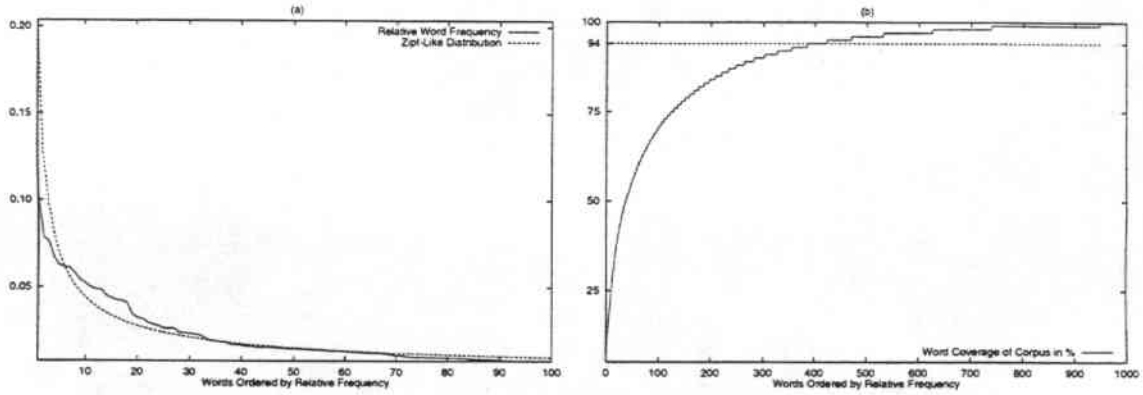

Figure 2.7: **The Word Distribution and Coverage:** *(a) The distribution of words in the training corpus follows Zipf's law. (b) Half of the words in the vocabulary cover more than 90 % of the corpus.*

and spoken English. But for a given corpus, especially for a collection of task-oriented dialogues as in our case, many words occurring with high frequency are salient for the dialogue or task at hand. These 'key' or 'trigger' words are the most important features for identifying arguments or speech acts by a statistical analysis, and should therefore be incorporated into the language model vocabulary.

A vocabulary that consists of the n most frequent words of a given corpus covers a large percentage of utterances of the corpus with a comparatively small number of words. Figure 2.7(b) illustrates this for the case of our training corpus, where a coverage of over 90 % is achieved with a vocabulary containing less than half of all different words in the corpus. A small vocabulary size also reduces the sparse data problem of the n-gram models.

For SALT we use a vocabulary size of 400 words (out of a total of 950 words); this corresponds to a coverage of 94 % of the 6594 utterances of the training corpus. Preliminary experiments on word-based speech act prediction show, that a coverage of roughly between 80 % and 90 % yields best results (see section 3.1).

All remaining words are clustered into linguistically defined word categories, often referred to as parts of speech (POS). This is done by a rule-based part of speech tagger[4]. A list of the part of speech tags used is given in table

[4]We use a modified version of Eric Brill's tagger [Bri92, Bri93, Bri94] and acknowledge Klaus Zechner for adopting it to spoken language.

A.2 in Appendix A.2.

Combining part of speech clustering with n-grams helps to cope with the problem of sparse data in language modelling. It also helps to increase the generalization capabilities of language models in general. Since out of vocabulary words are not all mapped to the same word category, this approach promises an information gain even for unknown words [NEK94, FLL+98, VR98, Rie99].

We should emphasize here, that the part of speech tagger is applied to all words in an utterance; this means, not only the out of vocabulary and unknown words are provided with a part of speech tag. Thus, the vocabulary consists in fact of the 400 most frequent word/tag pairs, and not words only. All remaining word/tag pairs are represented in the vocabulary by their part of speech tag only.

Accordance to their common usage, some words can appear in several different categories; the word '*one*' for example, can be tagged as a noun (`NN`), as a cardinal number (`CD`) or as an adjective (`JJ`). While multiple occurrences of a word in several different word/tag pairs increases the size of the corpus vocabulary, it can help in language modelling: different usages of a word are often related with a different meaning.

2.3.2 Segmentation and Labelling

Segmentation and labelling of utterances at speech act and argument level is accomplished by a multi-level classifier approach using statistical discourse grammars and likelihoods. The discourse grammars and likelihoods are modelled by n-gram back-off models, using an absolute discounting smoothing technique [NEK94, KN95, RSG97, CG98].

Running the Classifier

The basic idea is to use a multi-level hidden Markov model as shown in figure 2.5 in section 2.2.3 for speech act and argument detection.

Input to the multi-level classifier is the recognized utterance represented by a word string that has been pre-processed as described above. From this a lattice[5] of segments is produced by an A^* search procedure that can

[5] As commonly referred to in the speech recognition community, a lattice is the term used for a directed acyclic graph with a start node that can reach all nodes in the lattice, and an end node that can be reached from all nodes.

hypothesize segment boundaries and inserts every segment hypothesis into the segment lattice. The lookahead function of the search has been optimized to reflect the future effects of these segment boundaries.

The argument lattice is generated by replacing each segment by the set of all possible argument labels and assigning a likelihood for each segment given the argument. The segmentation and labelling at argument level is done in accordance with the argument part of the multi-level HMM. This requires the A^* search procedure to switch between argument specific n-gram models according to a hidden state while at the same time enforcing a discourse model on the hidden states.

The argument lattice is then segmented again and annotated with speech act labels. A Viterbi search can find the best combined argument and speech act sequence.

Training the Classifier

For training the multi-level classifier the speech act and argument labels are extracted from the parse-tree representation generated by the SOUP parser. A list of the speech act and argument labels used within SALT is given in the appendix A.3 in tables A.3 and A.4, respectively.

The extraction of speech act labels and segments from SOUP's output is straightforward: the speech acts are represented by the roots of the parse-trees. However, for extracting the argument labels and segments, we have to remark the following:

First, argument segments generally have a nested substructure — i.e. they constitute a sub-tree in the parse-tree representation of an utterance. Figure 2.8 shows an example for the nested representation of an argument of type '*super_time*'. Since we model a shallow parsing approach, we restrict our system to learn only the arguments that appear on top level in the parse-tree representation — i.e. the siblings of the root of each parse tree.

Second, the example in Figure 2.8 also shows, that not all words of an utterance are covered by arguments. As mentioned earlier, we have to introduce pseudo-arguments that capture the remaining words in order to make our two-level classifier approach work. Since the resulting sequence of arguments and pseudo-arguments is used to predict the underlying speech act label and segment, we chose to label the pseudo-arguments with the name of the speech act, as given by the SOUP parser.

For the purpose of grammatical inference several different arguments may

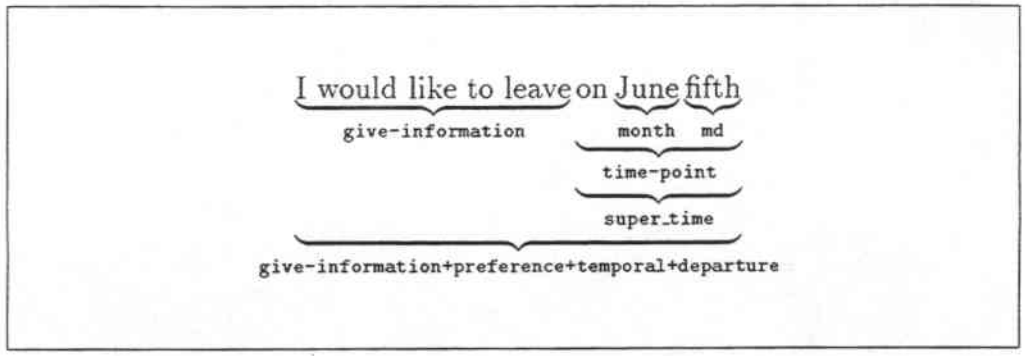

Figure 2.8: **The Nested Argument Substructure:** *Arguments in the parse-tree representation of an utterance can have a deeply nested substructure. The pseudo-argument* `give information` *is introduced to capture words not covered by* `super_time` *argument.*

exist that capture the same meaning or represent the same idea while still constituting different tokens in the SOUP grammar. For example, the top-level arguments `[gi:=availability=]`, `[ri:=availability=]`, `[=availability=]`, and `[be:=availability=]` all bear the meaning '*availability*'. Therefore, in such cases we map all argument names representing the same conceptual meaning into only one class. This helps to reduce the complexity of the model while improving consistency in our language models.

Another point to mention on training our HMM classifier with SOUP output is the handling of words, that are unknown to the SOUP parser or were skipped while parsing an utterance. Since we want to to have these words in our training data, we have to insert them appropriately into one of the argument or pseudo-argument segments. This is done one of the following ways:

If SOUP has detected no argument before skipping words — this includes the case where no argument is present in the current speech act — the words are inserted into a pseudo-argument with the name of the current speech act. Otherwise, the words are appended to the preceding argument segment.

2.3.3 Concept Prediction

The concept prediction is accomplished by an artificial neural network. The neural network we are using is a two-layer network without a hidden layer of neurons. The output layer is using a soft-max function [Jor95]. Training of

the neural net is done by use of the RPROP algorithm [RB94]. The implementation of the used variant of RPROP follows the one featured in SNNS [Zel93] and is part of the neural network library in the Janus recognition toolkit [ZFRW97, Rie99].

The task of the concept prediction is to estimate for each detected speech act segment the most probable combination of concepts based on the given speech act and its accompanying arguments. The speech act, predicted concepts and arguments shall then be mapped by the IF-mapper to a valid domain action, as specified by the C-STAR interchange format.

As expected, the information about the speech act and its accompanying arguments are the most salient features. However, experiments show, that a unigram of the words present in the given speech act segment can improve prediction performance significantly (see section 3.2). The word-unigram contains basically the same 400 most frequent words of the training corpus, which are used as the vocabulary of the language models for the HMM; except that for concept prediction no part of speech tags are used. This reveals, that in some cases a concept-combination is triggered merely by the presence of certain words in the given speech act segment.

Therefore, the input of the neural network consists of the speech act, its accompanying arguments and the word-unigram. Output of the neural network is one of the 141 different concept combinations that have been learned from the training corpus or an '*empty*' concept (accounting for domain actions that do not contain any concept). The list of concept combinations is given in table A.5 in appendix A.4.

Experiments on concept prediction with C4.5 [Qui93], a decision tree algorithm, and TiMBL [DZvv99], a memory based learner, showed results comparable to the proposed connectionist approach. For sake of simplicity of the resulting overall system we chose the connectionist approach, since the underlying neural network implementation is already part of the Janus recognition toolkit.

2.3.4 Argument Parsing

The grammar-based argument parsing stage produces the argument substructure that is needed to extend the shallow parse tree representation generated by the statistical analysis to a representation with full parse trees. This is done by running the SOUP parser on each detected argument segment and specifying a non-terminal start-symbol of the semantic grammar

that corresponds to the detected argument label[6].

The basic idea was originally to use special argument grammars that would only be capable of parsing the pre-segmented and labelled argument fragments as detected by the statistical analysis stage. Since a full semantic grammar is already available for our domain, we can achieve the same effect by stalling all grammar-rules that cannot be derived from the specified start-symbol. The advantage of argument grammars is that they are relatively trivial to design and write, compared to a full semantic grammar for parsing whole utterances in the given domain.

Since the argument labels used in the statistical analysis are not identical with the grammar's argument names, a mapping of argument labels to grammar argument names has to be carried out. This mapping follows basically the replacement scheme specified in table A.4 in appendix A.3.

As described in section 2.3.2 we cluster several grammar-arguments with the same conceptual meaning under one argument label. When parsing an argument segment the set of all grammar-arguments corresponding to the detected argument label has to be specified as start-points. The SOUP parser then produces several parse trees starting from any of the specified grammar tokens and returns the best parse according to internal heuristics. The SOUP output is then inserted into the shallow parse tree representation in place of the argument segments.

We have also implemented a fall-back mechanism, in case SOUP cannot parse an argument segment. Here, the argument label gets replaced by the most probable grammar argument name. This enables the IF-Mapper still to find an argument and map the given parse tree into a valid interchange format domain action.

2.3.5 Post-Processing

A post-processing step is needed, since the multi-level hidden Markov model used for segmenting and labelling utterances tends to over-generate — i.e. it predicts more arguments and domain actions than are actually present in the given utterance. Producing too many segments or segments that are too small, especially at the argument level, can result in invalid IF. We chose to implement a post-processing module at IF level, i.e. after the generated

[6]We acknowledge Marsal Gavaldà for creating the Soup parser and thank him for implementing the `parse_under` feature which does the trick of specifying a start token.

parse trees have been mapped by the IF-mapper, since at this level it is much easier to resolve parser errors or correct invalid or inconsistent parts of the IF representation.

The post-processing is done by an IF-level filter that basically eliminates empty[7], as well as repetitive arguments. Complete domain actions can also be removed from the final IF-representation, if it contains already a more detailed one with the same concept-combination.

With this approach we attack several problems at once: Empty arguments can have many causes with in the given system. They can be due to a wrong argument-level segmentation or labelling, such that salient words for the argument parser or the IF-mapper are missing and thus no argument-value can be extracted; or the SOUP parser could not parse the underlying argument segment, and the IF-mapper was unable to extract the value from the flat argument structure.

Repetitive arguments can stem from over-generative argument-level segmentation, but in this case usually one of the repetitive arguments is empty and gets removed anyway. The most common reason for repetitive arguments and domain actions are speaker idiosyncratic, such as false starts or phrase repetitions as shown in the two examples in figure 2.9.

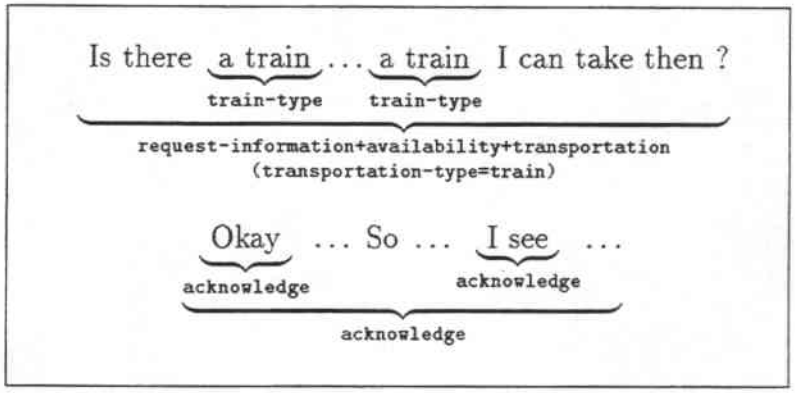

Figure 2.9: **The Post-Processing IF-Filter:** *Two examples showing elimination of a repetitive argument (above) and a repetitive domain action (below).*

It is interesting to note, that word or phrase repetitions are much safer

[7]We regard an argument as empty, when it has no associated argument-value in its IF-representation frame.

to detect and remove at IF-level than at word level. At word level it is not obvious how to differentiate wrong repetitions, that should be eliminated, from 'legal' ones, such as spelling or listing names or numbers.

2.4 System Implementation

The main part of the SALT system (ca. 1k lines of code) is written in Python, a highly efficient object oriented programming language for rapid prototyping. A Python library (ca. 10k lines of code independent from SALT) contains several modules that represent interfaces to the language modelling and neural network components of the Janus recognition toolkit. The Janus toolkit (ca. 100k lines of C-code accessible via a Tcl/Tk interface) also implements the A^* search and the speech recognizer (not used). The Python modules are hooked on top of Janus' Tcl/Tk interface, as Figure 2.10 demonstrates.

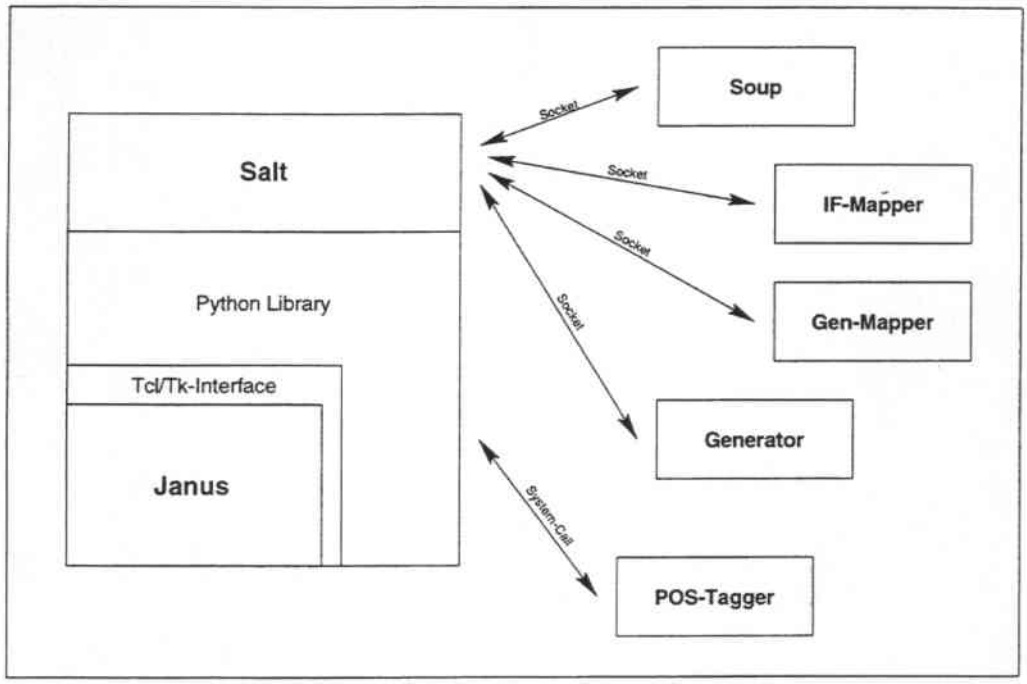

Figure 2.10: **The Implementation Architecture:** *Salt accesses the Janus toolkit, as well as all external components via Python modules.*

All external components are embedded into Python modules and classes,

either as socket connections (SOUP, IF-Mapper, Gen-Mapper, Generator) or via UNIX OS commands (part-of-speech tagger).

The training- and run-modules are short Perl scripts (ca. 300 lines of code), that call Python modules and routines. It is intended to port the training- and run-modules to Python, too, in an attempt to make the system implementation more homogeneous and to increase its portability and usability, e.g. as a socket version.

Chapter 3

Experiments and Results

This chapter describes the experiments we have conducted in order to show the feasibility of our approach, as well as helping us to resolve modelling and design issues during the creation of the SALT system.

We will first describe preliminary experiments on word-based speech act prediction, and compare our results to a similar study [FKWT98]. Then, results for experiments on concept prediction and multi-engine integration are reported. Finally, we will present results of an end-to-end system evaluation in the context of the C-STAR speech translation effort.

3.1 Word-Based Speech Act Prediction

Motivated by the experiments described in [FKWT98] we tried to reproduce the reported results.

The task is to estimate speech acts on an annotated English database on travel arrangements with 64 dialogues for training and 50 dialogues for testing. Experiments were conducted by use of the Janus recognition toolkit in Tcl/Tk; with pre-processing in Perl.

The approach reported in [FKWT98] uses 1-gram language models with domain dependent text-normalization, use of side information, and extensive heuristic tuning. The text-normalization maps several different words onto class expressions, such as 'numbers', 'hotel-name', etc. The reported result on word-based speech act prediction was 58.6 % correct.

In a first experiment we used the same domain dependent text-normalization as in [FKWT98]. Side information was also integrated, but no heuristic

tuning was needed to achieve similar or even better results. Instead a n-gram discourse model on the speech act sequence was applied. Results for different orders of the speech act discourse model, as well as the word-level n-gram model are summarized in table 3.1. The best result of 59.7 % correctly predicted speech acts was obtained by use of a 3-gram model at word-level combined with a bigram discourse model on the speech acts (see table 3.2).

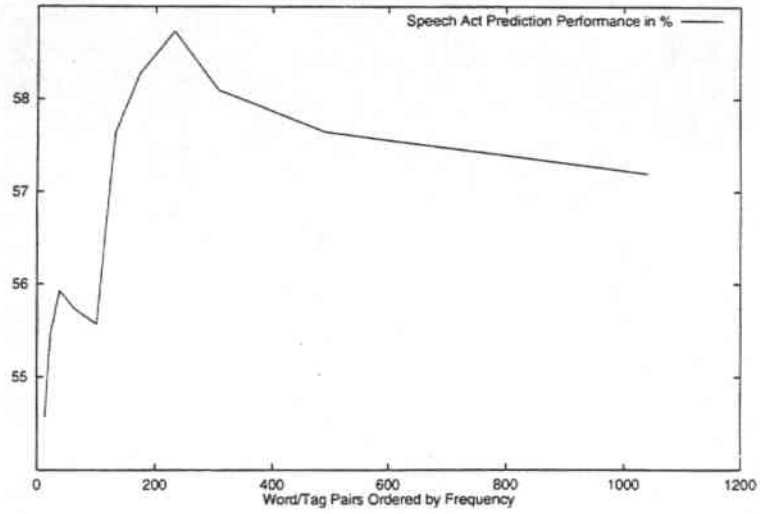

Figure 3.1: **Speech Act Prediction with Varying Vocabulary Size:** *Using a part of speech tagger we can map all but the n most frequent word/tag pairs into part of speech classes, thus varying the vocabulary size. The underlying discourse and language models are bigrams.*

In another experiment we used a part of speech tagger instead of the text-normalization to annotate the database. We mapped all but the n most frequent word/tag pairs on their part of speech tag. Figure 3.1 shows the prediction performance for different values of n. It can be seen, that using roughly between 200 and 300 word/tag pairs (out of 1042) in the vocabulary yields the best results — this corresponds to a vocabulary coverage of 80 % to 90 % of the training corpus.

In the experiment with varying vocabulary size the underlying discourse and language models were both bigrams. Table 3.1 displays results for different orders of the speech act discourse model, as well as the word-level n-gram model. The best result of 59.9 % correctly predicted speech acts was obtained by use of a 3-gram model at word-level combined with a bigram discourse

model on the speech acts plus tuning one parameter of the n-gram models (see table 3.2) — without tuning the best result of 58.7 % correct recognized speech acts was achieved with a bigram as discourse and language model.

word-level model	speech act model	with text-normalization	with pos-tagging
1-gram	1-gram	55.3 %	50.5 %
2-gram	1-gram	57.0 %	56.6 %
3-gram	1-gram	56.3 %	55.2 %
2-gram	1-gram	57.0 %	56.6 %
2-gram	2-gram	59.4 %	58.7 %
2-gram	3-gram	58.9 %	57.4 %

Table 3.1: **Speech Act Prediction Accuracy:** *Displayed is the percentage of correctly predicted speech acts using text-normalization and part-of-speech tagging. The results for part-of-speech tagging were achieved with a vocabulary of 230 word/tag pairs.*

Overall the above experiments show the feasibility of word-based speech act prediction by use of n-gram language and discourse models. Further we see, that bigram models are sufficient to achieve comparable — and sometimes even better — results as higher order n-gram models.

pre-processing	word-level model	speech act model	correct
n.a.	3-gram	2-gram	57.6 %
text-normalization	3-gram	2-gram	59.7 %
part-of-speech tagging	3-gram	2-gram	59.9 %
as reported in [FKWT98]: text-normalization	1-gram	n.a.	58.6 %

Table 3.2: **Pre-Processing for Speech Act Prediction:** *Displayed are best results on speech act prediction using different approaches for pre-processing. The results for part-of-speech tagging were achieved with heuristic tuning of one parameter, using a vocabulary of 230 word/tag pairs.*

Comparing the results in table 3.2 one can see, that no domain dependent text-normalization is needed to achieve comparable prediction results. Thus, we can reproduce, and even improve the results reported in [FKWT98] by a

more general approach using a domain-independent part of speech classifier. Our proposed approach is not only more general and robust in case of changes to the classifier-system, but it also requires no additional human modelling effort, once a part of speech tagger is available.

3.2 Concept Prediction

The task is to estimate the most probable combination of concepts for each detected speech act segment. We compare results from three different approaches: C4.5 [Qui93], a decision tree algorithm, TiMBL [DZvv99], a memory based learner, and the two-layer neural network described in section 2.3.3.

The experiments in this section serve two purposes: first, we want to find the features of the input representation that are salient for predicting the concepts; possible candidates are side information (sd), speech act label (sa), accompanying argument labels (arg), and a unigram of the words present in the given speech act segment (wrd). The word-unigram contains basically the same 400 most frequent words of the training corpus, that are used as the vocabulary of the language models in the HMM; except that for concept prediction no part of speech tags are used.

The results in table 3.3 state, that speech acts, arguments and the word-unigram are all salient features that improve the resulting prediction performance considerably; whereas the use of side information gives no advantage.

Input Features	C4.5	TiMBL
wrd (+sd)	66.2 % (65.1 %)	68.1 % (67.3 %)
sa, wrd (+sd)	70.8 % (71.4 %)	71.1 % (71.1 %)
arg, wrd (+sd)	74.9 % (74.1 %)	74.4 % (74.1 %)
arg, sa, wrd (+sd)	83.1 % (83.1 %)	82.6 % (82.6 %)

Table 3.3: **Salient Features for Concept Prediction:** *Displayed is the percentage of correctly predicted concepts, given the input features: arg - argument labels, sa - speech act label, sd - side information, and wrd - word-unigram. Values in brackets include side information as input feature.*

The second purpose of the experiments is to compare the performance of the different machine learning approaches, and find the best performing and most suitable one for integrating into the SALT system. Table 3.4 shows,

that the neural network outperforms both other approaches on the given test-set with the same input features, namely speech act, arguments and word-unigram. Since the components for implementing the neural network are part of the Janus recognition toolkit, which we already use for the language models, we chose to integrate the neural network approach of concept prediction into the framework of SALT.

Input Features	NN	C4.5	TiMBL
arg, sa, wrd	87.2 %	83.1 %	82.6 %

Table 3.4: **Comparing Models for Concept Prediction:** *The neural network model outperforms C4.5 and TiMBL. The input features are speech act, arguments, and word-unigram.*

3.3 End-to-End Evaluation

Within the C-STAR translation effort end-to-end system evaluations are performed on a regular basis. This involves a set of 200 to 300 unseen utterances in English that are transcribed and recognized by the Janus speech recognition system. Both sets (transcribed and recognized) are run through all system components, such that we get from translations into German and Japanese, and a paraphrase in English. The output utterances are then graded with respect to the original utterance according to three categories: perfect, ok, and bad.

A translation qualifies as perfect, if it looks 'natural' and 'smooth', and all information from the original utterance is transferred. In an 'ok' translation all information critical to the dialogue survives, but less relevant information can be missing. If critical information for the underlying dialogue is missing, the translation is considered to be bad.

For now, SALT has only been graded for English paraphrases. Table 3.5 gives the results of the last evaluation in comparison with the SOUP parser.

	Label	Recognition	Transcription
SALT:	perfect	31.7 %	36.9 %
	ok	18.8 %	20.4 %
	bad	49.5 %	42.7 %
	acceptable	50.5 %	57.3 %
SOUP:	perfect	39.4 %	47.6 %
	ok	22.2 %	26.1 %
	bad	38.4 %	26.3 %
	acceptable	61.6 %	73.7 %

Table 3.5: **C-STAR End-to-End Evaluation:** *Results for paraphrasing from English to English based on the average of two graders. The category 'acceptable' combines perfect and ok grades.*

3.4 Multi-Engine Integration

As described in section 2.2.2 we aim for the integration of SALT, SOUP and Pangloss into a multi-engine machine translation system within the framework of the Janus-III speech translation system. The underlying idea is to get an improved overall translation quality superior to the single systems.

Therefore, an 'oracle' experiment is conducted to estimate an upper bound of what can be achieved theoretically by a multi-engine integration. This is done by manually picking for each sentence of a given test set the best translation from all systems that shall be combined. The results for an oracle experiment on paraphrasing[1] recognized English sentences combining SALT and SOUP are displayed in table 3.6. It also shows oracle results for translating recognized sentences from English to German combining SALT, SOUP and Pangloss. The given results let hope for a large improvement by integrating the three systems into a multi-engine translation system.

Salt	Soup	Pangloss	oracle result
English to English paraphrase:			
47.3 % (30.3 %)	59.7 % (38.3 %)	n.a.	65.3 % (42.0 %)
English to German translation:			
50.4 % (23.3 %)	59.3 % (30.7 %)	64.6 % (28.3 %)	77.6 % (44.6 %)

Table 3.6: **Oracle Experiment Combining Salt, Soup and Pangloss:** *The displayed values are percentages of 'ok' grades based on recognized input utterances; the perfect grades are given in parentheses. The English to English oracle result combines Salt and Soup only.*

Motivated by these encouraging results a row of experiments was conducted to find a classifier for choosing the best German translation from one of the three systems. Basically three different approaches were tried:

The first approach makes use of SOUP's parse statistics, which are delivered together with each parse output. Among these, the coverage appears to be a salient feature for determining the quality of the parse output. The coverage denotes the number of parsed words, divided by the total number

[1] Paraphrasing means here 'translating' back into the same language, i.e. the output consists of English sentences that should reflect the meaning, and thus give a paraphrase, of the given input sentences.

of words in the utterance. A low coverage indicates that many words were skipped during the parse process and, therefore, the parse output may not represent all relevant information present in the given utterance. Using a threshold on the coverage can serve as a simple and effective classifier. For combining SOUP with Pangloss we achieve an absolute improvement of 2.1 % if we choose SOUP's output whenever it covers at least 95 % of all words in a sentence. Combining SOUP and SALT yields only a small improvement of 0.7 %; the threshold used here is 70 %. This approach is only applicable for combining SOUP with one additional system. It cannot be used for combining all three systems.

In the second approach statistical language models are applied as classifiers to judge which system ought to be used for translating a given utterance. The models base their decision on the sequence of parts of speech in the input utterance. Bigram back-off models are used to implement the classifiers. They are trained and evaluated on a set of 300 sentences that was annotated in the previous oracle experiment; the annotations identify for each input sentence the system that yields the best translation. Since the training set is comparatively small and no separate test set is available, training and testing is conducted in round-robin fashion. I.e., the set of 300 sentences is divided into ten sets of 30 sentences each; nine sets are used for training, while one set is held out to evaluate the classification performance. Thus, by rotating the training and test sets we can train and evaluate ten different classifiers. The overall number of correct classifications is the sum of the sentences classified correctly by each classifier on its respective test set. This approach yields only a small improvement in translation quality, e.g. in the case of combining SOUP and Pangloss only an improvement of 0.4 % is gained.

The third approach tries to combine the previous approaches in one classifier using an artificial neural network. We apply here the same type of neural network as implemented in the concept prediction module (see section 2.3.3). It allows each output unit to receive an additional input that can be determined by another knowledge source. This can be interpreted as a prior on the output distribution. It enables us to build a hybrid classifier using the coverage feature from the first approach as input while integrating the score of the statistical language models as additional knowledge source. As table 3.7 shows, the hybrid approach cannot improve the results obtained with the first approach. But, it allows integration of all three system into a multi-engine translation system that achieves an improvement in translation quality of 1.7 % over its best single components.

Translating English to German	oracle	coverage	lang. models & neural net
Soup & Salt	76.3 % (38.3 %)	60.0 % (31.4 %)	59.3 % (32.6 %)
Soup & Pangloss	75.0 % (39.0 %)	66.7 % (31.3 %)	65.3 % (30.3 %)
Soup & Salt & Pangloss	77.6 % (44.6 %)	n.a.	66.3 % (31.3 %)

Table 3.7: **Integrating Salt, Soup and Pangloss** *for translating from English to German. The displayed values are percentages of 'ok' grades based on recognized input utterances; the perfect grades are given in parentheses.*

However, it should be noted that the grammars used for generating German output from analyses with SALT or SOUP are in an early stage of development. They still lack a substantial amount of domain-coverage compared to the grammars used for generating English output. This is also stated by an experiment that combines SALT and SOUP for paraphrasing English sentences. Using the language model classifier from the second approach above (i.e. without the neural network) we almost achieve the results of the oracle experiment, which translates in an absolute improvement of 4.3 %. The results are given in table 3.8.

Salt	Soup	oracle	Salt & Soup
47.3 % (30.3 %)	59.7 % (38.3 %)	65.3 % (42.0 %)	64.3 % (40.0 %)

Table 3.8: **Integrating Salt and Soup** *for paraphrasing English sentences with language models as classifiers. The displayed values are percentages of 'ok' grades based on recognized input utterances; the perfect grades are given in parentheses.*

Chapter 4

Discussion

The main concern of this work is to design and implement an alternative parsing method, which performs shallow parsing of input utterances using a probabilistic framework, that allows derivation of model parameters automatically from a corpus, using statistical inference techniques.

The most prominent result of this work consists of the fact that the proposed approach of shallow statistical parsing is not only feasible in the context of a machine translation system for spoken language, but it is also practicable in the framework of a multi-engine machine translation system, which shows improved performance over the single integrated modules. Even a preliminary 'first shot' system implementing the proposed approach produces good results and performs relatively well compared to other well established systems, which have seen several man-years in development and improvement.

The rest of this chapter discusses the main contributions of our work for the area of natural language processing together with an outlook on possible ways to extend and improve the SALT system at its current state.

Hybrid Modelling

In fact the SALT system constitutes a hybrid approach of statistical and symbolic parsing. It combines shallow statistical parsing with a grammar-based parser for parsing the statistically detected sub-segments.

The statistical component makes the whole parse process more stable and robust, because it deals well with unanticipated input while the grammar-based component is very efficient and reliable in parsing small sub-segments of input in order to extract highly specific information, such as argument

values in our case.

Another advantage of this hybrid approach lies in the simplicity of the involved models: The statistical model can be restricted to one or more levels of the overall parsing process, which simplifies its implementation and training.

The grammar-based parsing module can basically be a very simple 'chunk-parser' that gets pre-segmented and labelled fragments of text-strings as input. The grammars involved in parsing the chunks are trivial, since they have to deal only with a very limited domain (specified by the fragment-label) and a very limited syntactic structure (fragments are only small parts of a sentence). They can be written in a fraction of time that a full grammar would need for being developed.

The hybrid model allows integration of additional information into the parse process. Cues such as speaker information, prosody, lip-reading, or other multi-modally recognized features can easily be utilized within the statistical framework. It can also be extended to use statistical discourse grammars for domain actions or concepts.

Practicability under Rigid Specifications

The integration of the SALT system within the framework of the Janus III machine translation system puts many restrictions on its design, specification and architecture:

The overall input/output function of the system is specified by the requirements of the embedding system. The actual features (speech acts, concepts and arguments) that shall be detected by the statistical analysis are pre-specified, and thus may not reflect an optimal choice for the machine learning task at hand.

Another point to mention is the quality of the training data generated with help of the SOUP parser. Since no correction mechanism in creating the training database is applied, its quality reflects the limited correctness of the SOUP parser — this means that roughly 30 to 40 percent of the used training examples are in fact erroneous parses.

Using the SOUP parser for generating training examples also requires implementation of the same word-level filter for the pre-processing stage as SOUP uses before parsing. This word-level filter was specifically designed for SOUP and its domain grammars, and thus may not be optimal in the pre-processing stage of a statistical parser.

In spite of the described disadvantages and drawbacks imposed by the integration into the Janus system and the need for generating training data automatically, the SALT system shows a good performance.

This states that our approach is practical even under the rigid specifications of an existing and highly developed machine translation system. And, the fact that the reported system was implemented in less than one man-year sheds a new light on the efficiency of statistical parsing approaches in general — particularly, if one thinks of the many ways our model could be extended or improved:

The SOUP parser could be used in reverse mode to generate training data from its grammars in cases, where grammar rules have already been implemented, but the coverage in the training database is low. The training database could also be manually searched for incorrect examples, and those could be deleted or — with considerably more effort — corrected.

One could also try to further cluster the learned arguments in order to reduce their number and conceptual overlap; or one could use different pseudo arguments or try clustering those. In learning the arguments and pseudo-arguments, the mapping of words into arguments could be changed.

Another idea would be to predict speech acts not only based on arguments, but based on words, too. Or one could implement an incremental prediction for speech acts and concepts — i.e. the whole combination of speech acts or concepts of a domain action is not predicted in one step, but each single speech act or concept step by step. The three last approaches could also improve the robustness of the speech act and concept prediction.

Robustness

Robustness is a critical feature for dealing with spontaneous speech, since it contains interrupts, restarts, ungrammaticalities or phrases with low information content (e.g. "*well*", "*let me see*", "*i dunno*", etc.), and can be corrupted with speech recognition errors. In the proposed approach the robustness is based on and inherent in its statistical framework. At each stage of the parse process a default or fall-back mechanism is implemented already within the statistical models.

If, e.g., an argument is not detected correctly, the speech act and concepts prediction can still be correct — even if no arguments are recognized at all. The same argument holds for speech acts and concepts. So, no matter how scrambled the input to the system might look like, it will still find the most

likely interpretation for it.

The principle of robustness even extends to the argument parsing and post-processing stage. In case, that the SOUP parser cannot parse one or more of the given argument segments, the IF-mapper will still get the labels of the detected arguments, speech acts and concepts. In most cases this is sufficient to produce a reasonable and — with the help of the IF-filter in the post-processing stage — correct IF-representation of the given utterance.

This differs by large from all known symbolic approaches, which have to model speech idiosyncrasies explicitly in their grammars or by 'soft' rules, that allow handling of exceptions during the parse process (e.g. skipping or filling in words).

However, at this point the translation system often does not profit from SALT's robustness. The IF-mapper, as well as the Gen-mapper and the generator were designed for the SOUP parser, which would either deliver a fully detailed parse-tree representation, or an empty one. Therefore, these modules cannot handle the defaulted or partial analysis SALT would generate in cases of scrambled input sentences.

Generality

Although SALT's specification is very rigid and specific, its design and implementation bear a large degree of generality:

The core components, statistical analysis and concept prediction, consist of language and domain independent modules that can easily be trained for other applications. For example, the input to the multi-level classifier can also be a lattice of words, e.g. produced by a speech recognizer. In fact, the multi-level hidden Markov model is also being used very successfully within the Clarity project for word-based prediction of discourse structures[1]; a task which traditional approaches have shown incapable of handling adequately [FLL+98, LRTGL99].

The part-of-speech tagger used in the pre-processing stage is a widely used domain-independent standard module, that is also available for languages other than English. The word-level filter is only necessary to work well with the training data supplied by SOUP; provided training data from another source, the filter can either be left out completely or one can try to implement a language and/or domain dependent filter helping to improve performance

[1]We acknowledge Klaus Ries for implementing the multi-level hidden Markov model, and thank him for making it accessible for our system.

(e.g. in the case of our domain a number pre-processing on word-level could help to differentiate between dates, account numbers, flight numbers, etc.). Thus, the only domain and language dependent part of the SALT system is the argument parsing stage. But as mentioned earlier, the argument grammars required here are relatively trivial to develop and implement.

On one hand, this large degree of generality makes it possible to port the system to a completely different task or even another language with only a small amount of work. In fact, given a sufficient amount of training data, only the argument grammar has to be exchanged in order to move the whole SALT system to another language.

On the other hand, the generality of the modules used within SALT leaves a great variety of possible improvements. For example, the part-of-speech tagger could be augmented by special domain or task dependent categories; or the word-level filter could be adapted to suit the language models better.

Economy

The most important advantage of the proposed approach lies in its economy considering time and man-power — and thus the total costs — for creating an analysis component for a speech translation system. As already mentioned before, the whole system was built in far less than one man-year (provided that the used modules were already available). And, porting it to a new language requires only a fraction of the previously invested effort of the development. In fact, the main work in building a new statistical parser involves only two tasks: writing a new argument grammar and creating an example-database for training of the statistical models.

The first task, writing new argument grammars, can be accomplished by a skilled grammar writer probably in matter of weeks. The second task, the generation of the training corpus, can be accomplished by several relatively unskilled annotators in parallel. Thus the proposed approach helps reduce total time and costs compared to traditional grammar-based approaches, where usually several highly skilled grammar-writers are occupied for months or even years to generate a suitable grammar.

Chapter 5

Conclusion

Traditionally, automatic natural language parsing and translation have been performed with various symbolic approaches. Many of these have the advantage of a highly specific output formalism, allowing fine-grained parse analyses and, therefore, very precise translations. Within the last decade, statistical techniques have been proposed for learning the parsing task in order to avoid the tedious manual modelling of grammars. Especially for parsing spontaneous speech a huge modelling effort is needed to cope with malformed, ungrammatical or (by recognizer errors) corrupted input. And, for each new language to be integrated in the translation system the same extensive modelling effort has to be invested, since no transfer of the grammar-inherent knowledge is possible.

The main goal of this work was to design and implement a shallow statistical analysis component for language translation that allows derivation of model parameters automatically from a corpus, using statistical inference techniques. The hybrid approach of combining the shallow statistical analysis with a grammar-based parser is highly portable and economical even in the domain of spoken language analysis: The required corpus for training the statistical models is relatively small (approximately 6,000 utterances), and only a minimum of hand modelling is required. The grammars used by the parser analyze parts of sentences only, thus they are relatively trivial and need not much effort in development.

The prototype system SALT was developed and tested in the context of the C-STAR II machine translation effort, a research consortium cooperating on speech translation for travel planning dialogues. It was implemented using modules from the Janus RTK, as well as other standard natural language

processing components. It consists of two main components: a shallow statistical analysis for detecting segments in input utterances, and a grammar-based parser for parsing the detected segments. The output of the system consists of an interlingua representation used and specified by the C-STAR consortium.

SALT is trained with transcribed and automatically (by a grammar-based parser) annotated sentences, and evaluated on both, transcribed and recognized, sentences. It performs relatively well — with respect to its prototypical state — compared to a grammar-based parser, which has seen a considerable effort in development and grammar-writing. A key result is obtained by using recognized data for evaluating the C-STAR system on English to German translation. With SALT the translation performance results in 50.4 % correctly translated sentences; the grammar-based parser achieves 59.3 % correct translations.

Although SALT's current performance does not reach the level of the grammar-based parser, experiments integrating both systems into a multi-engine translation system state an improved performance, superior over both single systems. An experiment on paraphrasing English sentences shows an absolute improvement of 4.3 %.

This work gives clear evidence for the successful application of statistical methods in the analysis of spontaneous speech. The proposed approach has clear advantages over traditional symbolic approaches even under rigid specifications.

The main advantages of the proposed approach lie in its robustness (inherent in the statistical models), its general and economic implementation (due to the use of mostly language and domain independent standard components), and its extensibility (the statistical models facilitate a simple integration of additional information, such as prosody, lip-reading, etc.).

Appendix A

Details of the Salt Architecture

A.1 Word-Level Filtering

Table A.1 lists the search strings together with their accompanying replacement strings used by the word-level filter in the pre-processing stage of the SALT system (see section 2.3.1).

Since the filter works on recognized word strings, the given strings contain special tokens used by the recognizer: token '<s>' denotes the start of an utterance, token '</s>' signals the end of an utterance, and the plus sign '+' is used instead of an apostrophe ' ' '.

Table A.1: **The List of Mapped Strings:**

Search String		**Replace String**
' any thing else '	→	' anything else '
' by any chance '	→	' '
' by the way '	→	' '
' will any of '	→	' will '
' would any of '	→	' would '
' do any of '	→	' do '
' does any of '	→	' does '
' have any of '	→	' have '
' has any of '	→	' has '
' are any of '	→	' are '

continued on next page

Table A.1: *continued*

Search String		Replace String
' is any of '	→	' is '
' there any of '	→	' there '
' will any '	→	' will '
' would any '	→	' would '
' do any '	→	' do '
' does any '	→	' does '
' has any '	→	' has '
' is any '	→	' is '
' how much '	→	' how_much '
' <s> i see '	→	' <s> i_see '
' <s> okay i see '	→	' <s> okay i_see '
' <s> well i see '	→	' <s> well i_see '
' i am '	→	' i -am- '
' am i '	→	' -am- i '
' am '	→	' a-m '
' -am- '	→	' am '
' ' '	→	' + '
' a couple a '	→	' a couple of '
' twelve o+clock noon '	→	' twelve pm '
' twelve o clock noon '	→	' twelve pm '
' twelve noon '	→	' twelve pm '
' noon '	→	' twelve pm '
' how+s it going '	→	' how is it going '
' how+s '	→	' how does '
' much better '	→	' better '
' ok '	→	' okay '
' that should give us '	→	' that way we have '
' diner+s '	→	' diners '
' ler+s '	→	' lers '
' +kay '	→	' okay '
' lets '	→	' let+s '
' let+s '	→	' let us '
' i+m '	→	' i am '

continued on next page

Table A.1: *continued*

Search String		Replace String
' +ll '	→	' will '
' +re '	→	' are '
' +d '	→	' would '
' +s '	→	' is '
' +ve '	→	' have '
' +till '	→	' till '
' +til '	→	' until '
' til '	→	' until '
' +bout '	→	' about '
' won+t '	→	' will not '
' can+t '	→	' can not '
' cannot '	→	' can not '
' n+t '	→	' not '
' doctor is '	→	' doctor+s '
' if not possible '	→	' '
' if possible '	→	' '
' if not '	→	' '
' also '	→	' '
' please do '	→	' _please_ do '
' <s> please </s> '	→	' <s> _please_ </s> '
' <s> then there '	→	' <s> there '
' <s> then </s> '	→	' <s> _then_ </s> '
' <s> then '	→	' <s> '
' bye then '	→	' bye '
' ok then '	→	' ok '
' okay then '	→	' okay '
' _then_ '	→	' then '
' please '	→	' '
' _please_ '	→	' please '
' <s> well </s> '	→	' <s> _well_ </s> '
' well enough '	→	' _well_ enough '
' well indeed '	→	' _well_ indeed '
' well in deed '	→	' _well_ in deed '

continued on next page

Table A.1: *continued*

Search String		Replace String
' <s> well '	→	' <s> '
' _well_ '	→	' well '
' as well </s> '	→	' </s> '
' actually '	→	' '
' basically '	→	' '
' already '	→	' '
' once '	→	' one time '
' twice '	→	' two times '
' thrice '	→	' three times '
' in fact '	→	' '
' where else '	→	' where '

A.2 Part of Speech Tagging

Table A.2 lists the part of speech tags used in the pre-processing stage of the SALT system (see section 2.3.1). Tagging is done by a modified version of Eric Brill's rule-based part of speech tagger [Bri92, Bri93, Bri94], which was adopted to spoken language by Klaus Zechner.

Table A.2: **The List of Part of Speech Tags:**

Tag	Part of Speech
AFF	affirmative particle
ANA	anaphoric element
AUX	auxiliary
CC	conjunction
CCC	conjunction, constituent
CD	cardinal number
CV	conversational words
DT	determiner
EX	explicative
JJ	adjective
JJR	adjective, comparative
JJS	adjective, superlative
NEG	negation particle
NN	noun
NNP	proper noun
NNPS	proper noun, plural
NNS	noun, plural
PREP	preposition
PRP	personal pronoun
PRP$	personal pronoun, possessive
PRPA	personal pronoun, accusative case
RB	adverb
RBR	adverb, comparative
RBS	adverb, superlative
RP	verb particle
TO	'to' + infinitive

continued on next page

Table A.2: *continued*

Tag	Part of Speech
UH	colloquials
VB	verb, infinitive
VBD	verb, past
VBG	verb, gerund
VBN	verb, past participle
VBP	verb, present
VBZ	verb, inflection (3rd sgl. present)
WDT	relative pronoun
WP	wh-particle, attributive
WRB	wh-particle, isolated

A.3 Segmentation and Labelling

Table A.3 lists the speech act labels used in the statistical analysis stage of the SALT system.

Table A.3: **The List of Speech Act Labels:**

Speech Acts
accept
acknowledge
acknowledge-action
affirm
affirm-action
agent:request-information
apologize
client:request-action
closing
delay-action
end-action
give-information
greeting
greeting-nice-meet
greeting-request
greeting-response-bad
greeting-response-good
introduce-self
negate
negate-give-information
negate-request-action
negate-testing-ready
not-understand
offer
offer-information
offer-reservation
please-wait
reject

continued on next page

Table A.3: *continued*

Speech Acts
request-action
request-affirmation
request-delay-action
request-information
request-introduce-self
request-neg-affirmation
request-repeat
request-suggestion
request-verification
request-verification-delay-action
request-verification-end-action
request-verification-give-information
request-verification-request-action
suggest
testing
testing-present
testing-problem
thank
uncertainty
verify
verify-give-information
verify-request-action
verify-request-information
welcome

Table A.4 lists the argument labels used in the statistical analysis stage of the SALT system together with the argument names as they appear in the parse tree representation of SOUP's output. We clustered arguments denoting the same conceptual meaning into one argument label, respectively.

Table A.4: *continued*

Argument Label	Corresponding Soup-Arguments
	[q:frequency=]
hotel-name_question	[hotel-name_question]
hotel-name_unknown	[hotel-name_unknown=]
hotel-type	[q:hotel-type=]
how-many	[how-many=]
how-many_question	[how-many_question]
include_unknown	[include_unknown=]
letters	[letters=]
location	[q:location=]
	[time:location=]
	[where-q:location=]
method	[q:method=]
numeral	[q:numeral=]
occupancy	[occupancy=]
origin	[between:origin=]
	[intro-prep:origin=]
	[origin=]
	[prep:origin=]
origin_question	[origin_question]
origin_unknown	[origin_unknown=]
per-unit	[per-unit=]
person-name	[person-name=]
	[unk-no-title:person-name=]
person-name_question	[person-name_question]
price	[price-req-q:price=]
	[q:price=]
price-difference	[=price-difference=]
price-range	[price-range=]
price-type	[price-type=]
purpose	[purpose=]
rate	[rate=]
room-location	[room-location=]
room-number	[room-number=]

continued on next page

Table A.4: *continued*

Argument Label	Corresponding Soup-Arguments
room-type	[q:room-type=]
room-view	[room-view=]
row-number	[row-number=]
season	[season=]
seat-type	[seat-type=]
sight-type_question	[sight-type_question]
super_bed-type	[super_bed-type=]
super_carrier-name	[super_carrier-name=]
super_contain	[super_contain=]
super_destination	[and:super_destination=]
	[super_destination=]
super_flight-number	[super_flight-number=]
super_flight-type	[super_flight-type=]
super_for-whom	[for:super_for-whom=]
	[super_for-whom=]
super_hotel-facility	[super_hotel-facility=]
super_hotel-name	[super_hotel-name=]
super_hotel-service	[super_hotel-service=]
super_hotel-type	[super_hotel-type=]
super_include	[super_include=]
super_location	[prep:super_location=]
	[super_location=]
super_method	[super_method=]
super_numeral	[super_numeral=]
super_price	[super_price=]
super_room-type	[super_room-type=]
super_sight-name	[super_sight-name=]
super_sight-type	[super_sight-type=]
super_time	[super_time=]
super_transportation-type	[super_transportation-type=]
super_via	[super_via=]
super_weather	[good:super_weather=]
	[noun:super_weather=]

continued on next page

Table A.4: *continued*

Argument Label	Corresponding Soup-Arguments
	[super_weather=]
super_who	[party:super_who=]
	[super_who=]
super_x-predicate	[super_x-predicate=]
super_x-subject	[super_x-subject=]
telephone-number	[telephone-number=]
temperature	[temperature=]
time	[exp:time=]
	[q:time=]
	[time=]
time-relativity	[time-relativity=]
time_question	[what:time_question]
to-whom	[greet:to-whom=]
tour-type	[tour-type=]
train-name	[train-name=]
train-type	[train-type=]
train-type_question	[train-type_question]
trip-type	[trip-type=]
unavailability	[=unavailability=]
	[gi:=unavailability=]
weather_question	[weather_question]
web-page-image	[web-page-image=]
web-page-information	[web-page-information=]
web-page-object	[web-page-object=]
what	[num:what=]
	[what=]
	[write-num:what=]
	[write:what=]
with-how-many	[with-how-many=]
with-whom	[with-whom=]
x-car-type	[x-car-type=]

A.4 Concept Prediction

Table A.5 lists all concept combinations that have been learned by the neural network in the statistical analysis stage of SALT from the training corpus.

Table A.5: **The List of Concept Combinations:**

empty
+availability
+availability+hotel
+availability+room
+availability+web-page
+budget
+cancellation+penalty
+cancellation+price+transportation
+cancellation+temporal
+change+features+flight
+change+penalty
+change+reservation
+change+temporal
+click+features+web-page
+confirmation
+confirmation+features+flight
+confirmation+features+hotel
+confirmation+features+room
+confirmation+features+transportation
+confirmation+numeral
+confirmation+payment
+confirmation+temporal+room
+display+confirmation+numeral
+display+confirmation+numeral+checkin
+display+features+web-page
+expiration-date
+features
+features+activity
+features+admission

continued on next page

Table A.5: *continued*

+features+arrival
+features+arrival+flight
+features+arrival+train
+features+arrival+transportation
+features+attraction
+features+departure
+features+departure+flight
+features+departure+train
+features+departure+transportation
+features+event
+features+flight
+features+hotel
+features+party
+features+room
+features+stay
+features+tour
+features+train
+features+transportation
+features+trip
+features+web-page
+features+x-car-rental
+help
+help-again
+help-later
+location
+location+hotel
+location+room
+location+stay
+name
+neg-preference+features+x-car-rental
+numeral
+order+features+admission
+payment
+possibility

continued on next page

Table A.5: *continued*

+preference
+preference+features
+preference+features+admission
+preference+features+attraction
+preference+features+event
+preference+features+flight
+preference+features+hotel
+preference+features+room
+preference+features+stay
+preference+features+tour
+preference+features+train
+preference+features+transportation
+preference+features+trip
+preference+features+x-car-rental
+preference+location
+preference+location+hotel
+preference+location+room
+preference+price
+preference+price+hotel
+preference+temporal
+preference+temporal+arrival
+preference+temporal+departure
+preference+temporal+hotel
+preference+temporal+room
+preference+temporal+stay
+preference+temporal+trip
+price
+price+hotel
+price+refund
+price+room
+price+trip
+price-fluctuation
+price-fluctuation+room
+purchase+features+admission

continued on next page

Table A.5: *continued*

+reservation
+reservation+features
+reservation+features+attraction
+reservation+features+flight
+reservation+features+hotel
+reservation+features+room
+reservation+features+train
+reservation+features+transportation
+reservation+features+trip
+reservation+features+x-car-rental
+reservation+location+hotel
+reservation+location+room
+reservation+name
+reservation+numeral
+reservation+price+room
+reservation+temporal
+reservation+temporal+departure
+reservation+temporal+hotel
+reservation+temporal+room
+reservation+temporal+trip
+search
+search+features
+search+features+flight
+search+features+room
+spelling
+telephone-number
+temporal
+temporal+arrival
+temporal+checkin
+temporal+checkout
+temporal+departure
+temporal+event
+temporal+hotel
+temporal+minimum-stay

continued on next page

Table A.5: *continued*

+temporal+room
+temporal+stay
+temporal+trip
+time
+time-difference
+view
+view+features+web-page
+weather
+write
+x-predicate
+x-take-with

Bibliography

[ALP66] Automatic Language Processing Advisory Committee ALPAC. Languages and machines: Computers in translation and linguistics. Technical Report 1416, National Academy of Sciences, Washington, D.C., 1966.

[BCD+90] P. Brown, J. Cocke, S. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16:79–85, 1990.

[Ber91] George Berg. Learning recursive phrase structure: Combining the strengths of pdp and x-bar syntax. Technical Report TR 91-5, Department of Computer Science, University at Albany, State University of New York, 1991.

[BPW94] Finn Dag Buø, Thomas Polzin, and Alex Waibel. Learning complex output representations in connectionist parsing of spoken language. In *Proceedings of the ICASSP*. IEEE, 1994.

[Bri92] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing (ACL-92)*, Trento, Italy, 1992.

[Bri93] Eric Brill. *A Corpus-Based Approach to Language Learning.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1993.

[Bri94] Eric Brill. Some advances in part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, WS, 1994.

[Buø96] Finn Dag Buø. *FeasPar - A Feature Structure Parser Learning to Parse Spontaneous Speech.* PhD thesis, Universität Karlsruhe, Fakultät für Informatik, Karlsruhe, Germany, July 1996.

[BW96a] Finn Dag Buø and Alex Waibel. Feaspar : A feature structure parser learning to parse spoken language. In *Proceedings of the COLING 96*, Kopenhagen, 1996.

[BW96b] Finn Dag Buø and Alex Waibel. Learning to parse spontaneous speech. In *Proceedings of the ICSLP 96*, Philadelphia, PA, 1996.

[BW96c] Finn Dag Buø and Alex Waibel. Search in a learnable spoken language parser. In *Proceedings of the ECAI 96*, Budapest, 1996.

[CG98] Stanley F. Chen and Joshua T. Goodman. An empirical study of smoothing techniques for language modelling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, Cambridge, Massachusetts, August 1998.

[DZvv99] Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. Timbl: Tilburg memory based learner. ILK Technical Report ILK 99-01, Induction of Linguistic Knowledge, Computational Linguistics, Tilburg University, Tilburg, Netherlands, January 1999.

[FKWT98] Toshiaki Fukada, Detlef Koll, Alex Waibel, and Kouichi Tanigaki. Probabilistic dialogue act extraction for concept based multilingual translation systems. In *Proceedings of the ICSLP 98*, Sydney, Australia, 30th November – 4th December 1998.

[FLL+98] Michael Finke, Maria Lapata, Alon Lavie, Lori Levin, Laura Mayfield Tomokiyo, Thomas Polzin, Klaus Ries, Alex Waibel, and Klaus Zechner. Clarity: Automatic discourse and dialogue analysis for a speech and natural language processing system. In *Proceedings of the AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, March 1998.

[FNF+94] R. Frederking, S. Nirenburg, D. Farwell, S. Helmreich, E. Hovy, K. Knight, S. Beale, C. Domashnev, D. Attardo, D. Grannes,

and R. Brown. Integrating translations from multiple sources within the pangloss mark iii machine translation. In *Proceedings of the First Conference of the Association for Machine Translation in the Americas (AMTA-94)*, Columbia, Maryland, 1994.

[FRH97] R. Frederking, A. Rudnicky, and C. Hogan. Interactive speech translation in the diplomat project. In *Proceedings of the Spoken Language Translation Workshop at the 35th Meeting of the Association for Computational Linguistics (ACL-97)*, Madrid, Spain, 1997.

[Gav98] Marsal Gavaldà. The SOUP home page. `http://www.is.cs.cmu.edu/ISL.speech.parsing.soup.html`, June 1998.

[GN91] Kenneth Goodman and Sergei Nirenburg, editors. *The KBMT Project: A Case Study in Knowledge-Based Machine Translation.* Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[GSB+95] P. Geutner, B. Suhm, F. D. Buø, T. Kemp, A. Lavie, L. Mayfield, A. E. McNair, I. Rogina, T. Sloboda, W. Ward, M. Woszczcyna, and A. Waibel. Integrating different learning approaches into a multilingual spoken translation system. In *Workshop on New Approaches to Learning for Natural Language Processing*, Montreal, Canada, August 1995. International Joint Conference on Artificial Intelligence.

[GW98] Marsal Gavaldà and Alex Waibel. Growing semantic grammars. In *Proceedings of COLING/ACL-98*, 1998.

[Hou98] Juliane House. *Routledge Encyclopedia of Translation Studies*, chapter Quality of Translation, pages 197–200. Routledge, London, U.K., 1998.

[Jai91] Ajay N. Jain. *PARSEC: A Connectionist Learning Architecture for Parsing Spoken Language.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, December 1991.

[JBC+97] Daniel Jurafsky, Rebecca Bates, Noah Coccaro, Rachel Martin, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Andreas

Stolcke, Paul Taylor, and Carol Van Ess-Dykema. Automatic detection of discourse structure for speech recognition and understanding. In *IEEE Workshop on Speech Recognition and Understanding*, September 1997.

[Jor95] Michael Jordan. Why the logistic function? a tutorial discussion on probabilities and neural networks. `ftp://psyche.mit.edu/pub/jordan/uai.ps.Z`, 1995.

[KN95] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of ICASSP-95*, 1995.

[LGLW98] Lori Levin, Donna Gates, Alon Lavie, and Alex Waibel. An interlingua based on domain actions for machine translation of task-oriented dialogues. In *Proceedings of the ICSLP 98*, Sydney, Australia, 30th November - 4th December 1998.

[LLZ+97] A. Lavie, L. Levin, P. Zhan, M. Taboada, D. Gates, M. Lapata, C. Clark, M. Broadhead, and A Waibel. Expanding the domain of a multi-lingual speech-to-speech translation system. In *Proceedings of the Workshop on Spoken Language Translation*, Madrid, Spain, July 1997. ACL/EACL.

[LRTGL99] Lori Levin, Klaus Ries, Ann Thymé-Gobbel, and Alon Lavie. Tagging of speech acts and dialogue games in spanish call home. In *Proceedings of the AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, 1999.

[MD91] Risto Miikulainen and Michael G. Dyer. Natural language processing with modular pdp networks and distributed lexicon. *Cognitive Science*, 15:343–399, 1991.

[MGS+95] L. Mayfield, M. Gavaldà, Y.-H. Seo, B. Suhm, W. Ward, and A. Waibel. Parsing real input in janus: A concept based approach. In *Proceedings of TMI-95*, 1995.

[NEK94] Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modelling. *Computer, Speech, and Language*, 8:1–38, 1994.

[Nid64] Eugene A. Nida. *Toward a Science of Translating.* E. J. Brill, Leiden, 1964.

[NM94] M. Nagata and T. Morimoto. First steps towards statistical modeling of dialogue to predict the speech act type of the next utterance. *Speech Communication*, 15:193–203, 1994.

[OAM+92] L. Osterholtz, C. Augustine, A. McNair, I. Rogina, H. Saito, T. Sloboda, J. Tebelskis, A. Waibel, and M. Woszczyna. Testing generality in janus: A multi-lingual speech translation system. In *Proceedings of ICASSP.* IEEE, 1992.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *IEEE ASSP Magazine*, pages 267–296, 1989.

[RB94] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE Int. Conf. on Neural Networks*, volume 15, pages 193–203, 1994.

[REKK96] N. Reithinger, R. Engel, M. Kipp, and M. Klesen. Predicting dialogue acts for a speech-to-speech translation system. In *ICSLP*, 1996.

[Rie94] Klaus Ries. Korpusbasierte techniken zum lernen von übersetzung spontan gesprochener sprache. Master's thesis, Universität Karlsruhe, Fakultät für Informatik, Karlsruhe, Germany, July 1994.

[Rie99] Klaus Ries. Hmm and neural network based speech act detection. In *ICASSP-99*, Phoenix, Arizona, March 1999.

[RJ86] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

[RK97] Norbert Reithinger and Martin Klesen. Dialogue act classification using language models. In *Proceedings of EuroSpeech-97*, pages 2235–2238, Rhodes, Greece, 1997.

[RSG97] Klaus Ries, Bernhard Suhm, and Petra Geutner. Language modelling in janus. internal document of the Interactive Systems Laboratories at Carnegie Mellon University, October 1997.

[RV91] Katarina Reiss and Hans J. Vermeer. *Grundlegung einer allgemeinen Translationstheorie.* Number 147 in Linguistische Arbeiten. Niemeyer, Tübingen, 2nd edition, 1984/1991.

[Sch93] Hinrich Schütze. Translation by confusion. In *Spring Symposium on Machine Translation.* AAAI, 1993.

[Som98] Harold L. Somers. *Routledge Encyclopedia of Translation Studies*, chapter Machine Translation, pages 136–149. Routledge, London, U.K., 1998.

[SRC+99] Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. Dialog act modelling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 1999. to appear.

[SSB+98] Andreas Stolcke, Elizabeth Shriberg, Rebecca Bates, Noah Coccaro, Daniel Jurafsky, Rachel Martin, Marie Meteer, Klaus Ries, Paul Taylor, and Carol Van Ess-Dykema. Dialog act modeling for conversational speech. In *AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, March 1998.

[TKI+97] Paul Taylor, Simon King, Stephen Isard, Helen Wright, and Jacqueline Kowtko. Using intonation to constrain language models in speech recognition. In *EUROSPEECH*, Rhodes, Greece, 1997.

[VR98] Carol Van Ess-Dykema and Klaus Ries. Linguistically engineered tools for speech recognition error analysis. In *Interna-*

tional Conference on Spoken Language Processing (ICSLP-98), Sidney, Australia, 30th November – 4th December 1998.

[Wai96] Alex Waibel. Interactive translation of conversational speech. *Computer*, 7(29), July 1996.

[WAWB+94] M. Woszczyna, N. Aoki-Waibel, F. D. Buø, N. Coccaro, K. Horiguchi, T. Kemp, A. Lavie, A. McNair, T. Polzin, I. Rogina, C. P. Rose, T. Schultz, B. Suhm, M. Tomita, and A. Waibel. Janus 93: Towards spontaneous speech translation. In *Proceedings of ICASSP*, pages 345–348, Adelaide, Australia, April 1994. IEEE.

[WBG+98] Monika Woszczyna, Matthew Broadhead, Donna Gates, Marsal Gavaldà, Alon Lavie, Lori Levin, and Alex Waibel. A modular approach to spoken language translation for large domains. In *Proceedings of the 1998 Conference of the Association for Machine Translation in the Americas (AMTA-98)*, 1998.

[WJM+91] A. Waibel, A. N. Jain, A. E. McNair, H. Saito, A. G. Hauptmann, and J. Tebelskis. Janus: A speech-to-speech translation system using connectionist and symbolic processing strategies. In *Proceedings of ICASSP*. IEEE, May 1991.

[WJM+92] A. Waibel, A. N. Jain, A. E. McNair, J. Tebelskis L. Osterholtz, H. Saito, O. Schmidbauer, T. Sloboda, and M. Woszczyna. Janus: Speech-to-speech translation using connectionist and non-connectionist techniques. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems*, volume 7. Morgan Kaufmann Publishers, 1992.

[WKNN97] V. Warnke, R. Kompe, H. Niemann, and E. Nöth. Integrated dialog act segmentation and classification using prosodic features and language models. In *Eurospeech*, pages 207–210, 1997.

[Wor98] Karsten Worm. A model for robust processing of spontaneous speech by integrating viable fragments. In *Proceedings of COLING-ACL-98*, Montreal, Canada, August 1998.

[Wos] Monika Woszczyna. The C-STAR II Homepage. http://www.is.cs.cmu.edu/cstar/.

[WW94] Stefan Wermter and Volker Weber. Learning fault-tolerant speech parsing with screen. In *Proceedings of the NCAI-94*, Seattle, WS, 1994.

[Zel93] Andreas Zell. SNNS user manual, version 3.0. Technical Report 3, University of Stuttgart, Institute for Parallel and Distributed High Perfomance Systems, 1993.

[ZFRW97] Torsten Zeppefeld, Michael Finke, Klaus Ries, and Alex Waibel. Recognition of conversational telephone speech using the janus speech engine. In *Proceedings of the ICASSP'97*, München, Germany, 1997.

[Zip29] George K. Zipf. Relative frequency as a determinant of phonetic change. In *The Havard Studies in Classical Philiology*, volume Volume XL. Havard Press, 1929.