Universität Zürich[UZH]

Institut für Computerlinguistik

Integration von Finite-State Transducer-Technologien in Apertium zur Maschinellen Übersetzung morphologisch komplexer Sprachen

Seminararbeit im Seminar
«Maschinelle Übersetzung»
Herbstsemester 2011

Vorgelegt von
Hernani Marques

15. Juni 2012

Betreuer
Anne Göhring
Magdalena Jitca
Prof. Dr. Michael Hess

Zusammenfassung

Diese Seminararbeit liefert einen Überblick über die **Apertium**-Plattform zur *Maschinellen Übersetzung* - mit besonderem Augenmerk auf die morphologische Analyse- und Generierungsphase der Transfer-Pipeline, in der es möglich ist das Helsinki Finite-State Technology-Framework (**HFST**) zur Anwendung zu bringen, um Morphologien von Sprachen zu verwalten, die mit linguistischer Systematik rein konkatenativ (mit den **Apertium**-Bordmitteln) nur schwer zu erfassen sind. Diskutiert wird die Integration am Beispiel des Sprachpaars *sme-nob* (Nordsamisch zu Norwegisch), das mit Abschluss dieser Arbeit den Einzug in die stabilen (produktiv-nutzbaren) Sprachpaare von **Apertium** geschafft hat.

Inhaltsverzeichnis

1 Einführung

1.1 Motivation und Thema

Im Rahmen des Seminars "Maschinelle Übersetzung" bei Prof. Dr. Michael Hess, Anne Göhring und Magdalena Jitca im Herbstsemester 2011 haben wir uns mit verschiedenen Aspekten der automatischen Übersetzung natürlichsprachlicher Texte beschäftigt, darunter auch mit Systemen, welche regelbasiert (*RBMT*) und anderen, die statistikbasiert (*SBMT*) arbeiten. Zusammen mit Simon Hafner habe ich mich damals dazu entschieden zwei Systeme mit einem Vortrag zu beleuchten, welche frei im Quellcode verfügbar und nutzbar sind: **Apertium** als *RBMT*- und **Moses** als *SBMT*-System. Insbesondere haben wir die Vorteile von *FOSS-MÜ*-Systemen [1] für Minderheitensprachen beleuchtet. In meinem Teil zu **Apertium** habe ich das Gesellschaftliche zu stark gewichtet, so dass das Technische dementsprechend auf der Strecke geblieben ist. Mit dieser Seminararbeit möchte ich das nun ausräumen. Ich fokussiere mich auf **Apertium** als eine offene Plattform, und wie sie genutzt werden kann, um mit Sprachen umzugehen, die eine komplexe Morphologie aufweisen.

Was soll gezeigt werden?

Apertium ist ein *RBMT*-System, das insbesondere dafür bekannt ist mittels *Shallow-Transfer* zwischen linguistisch ähnlichen Sprachpaaren zu übersetzen. Bei den ursprünglich involvierten Sprachen handelt es sich darüber hinaus um solche, die über eine (vergleichsweise) einfache Morphologie aufweisen und indoeuropäischen Ursprungs sind. Im Zuge nun der Übersetzung morphologisch komplexer und weiter voneinander entfernten Sprachpaaren kann das Helsinki Finite- State Technology - Framework (HFST) eingesetzt werden; auf das wird insbesondere im Kapitel 2.2 eingegangen. Das Framework bietet Schnittstellen zu bestehenden Finite-State Transducer-Technologien (z. B. **Foma** oder **OpenFST**) an und ermöglicht somit eine Wiederverwendung

1. MÜ-*Systeme, die Free and Open Source Software sind*

(morphologischer) sprachtechnologischer Ressourcen für den Bereich der *Maschinellen Übersetzung* zwischen Sprachen, von der zumindest eine geprägt ist von einem agglutinierenden oder stark flektierenden Sprachbau.

In meiner Seminararbeit möchte ich also zunächst aufzeigen, was **Apertium** und **HFST** sind (in Kapitel 2), und dann wie sie zueinander stehen und integriert werden können , um mehr Sprachpaare übersetzen zu können. Diese Integration mit **HFST** theoretisch und empirisch aufzudecken, ist Inhalt von Kapitel 3. Die entsprechenden Illustrationen liefere ich anhand des Sprachpaars *sme-nob* (Nordsamisch zu Norwegisch), das zum einen gut dokumentiert ist und mir zum anderen einen Zirkelschluss zu meinem Ursprungsthema des Seminars erlaubt, nämlich: Aufzuzeigen, wie **Apertium** dazu genutzt werden kann Minderheitensprachen zu unterstützen - nur diesmal bedeutend mehr in der technischen Beleuchtung der Sache. Interessant an diesem Sprachpaar ist zudem, dass eine der involvierten Sprachen, nämlich das Nordsamische, morphologisch komplex ist *und* mit dem Norwegischen wenig gemein hat , d. h. das Sprachpaar ist von hoher linguistischer Distanz. Das ist ein Bereich für den die **Apertium**-Plattform ursprünglich nicht konzipiert wurde. In einem Überblick zu zeigen, wie mit diesen Anforderungen an das *RBMT*-System umgegangen wird, ist Ziel meiner Arbeit.

Kapitel 4 - der Schluss - setze ich mit einer Zusammenfassung der behandelten Materie um und stelle die wichtigsten Erkenntnissen meiner Seminararbeit in den Raum. Auch ist das der Platz, um einen Blick in die Zukunft zu wagen, was in **Apertium** in der nächsten Zeit einfliessen wird. In dem Zusammenhang sind Chatgespräche und Hinweise, die ich von (wohl bekannten) **Apertium**-Entwicklern erhalten habe, wertvoll.

1.2 Verwendete Umgebung und Konfiguration

Alle meine hier beschriebenen Versuche habe ich auf einem amd64-System durchgeführt, auf das die stabile **Debian**-Veröffentlichung 6.0.5 ("Squeeze") [2] installiert ist. Während **Apertium** bei vielen Distributionen von GNU/Linux paketiert vorliegt, gilt das für das **HFST**-Framework nicht; dieses wird allerdings benötigt, um mit morphologisch komplexen Sprachen zu arbeiten.

Will man **Apertium** zunächst aber nur in den stabilen und für die Veröffentlichung herausgegebenen Sprachpaaren testen, so kann unter **Debian** das Paket *apertium* installiert werden. Zur Installation der Dateien für eine konkrete (stabile) Übersetzungsrichtung, wie z. B. *fr-ca* (von Französisch nach Katalanisch), muss zusätzlich das Paket *apertium-fr-ca* installiert werden. In diesem Paket sind die effektiven Lexika in binärer Fassung für diese Sprachrichtung, einschliesslich den Transferregeln und anderen benötigten Dateien, enthalten.

2. **Debian**-Webseite: `http://www.debian.org/News/2012/20120512` (letzter Zugriff: 08.06.2012)

Ist das geschehen, kann **Apertium** auf der Kommandozeile wie folgt getestet werden:

```
$ echo "J'aime manger du chocolat." | apertium fr-ca #1
Estima menjar de la xocolata.
$ echo "Ils aiment manger du chocolat." | apertium fr-ca #2
Estimen menjar de la xocolata.
```

Beide Sätze #1 [3] und #2 [4] zeigen auf, dass **Apertium** bei (einfachen) Sätzen der vorgezeigten Übersetzungsrichtung korrekt arbeitet.

Für weitergehende Versuche, insbesondere mit experimentellen Sprachpaaren, ist mit den vorkompilierten Paketen aus dem **Debian**-Paketsystem wenig zu erreichen. Für die meisten der neuesten Sprachpaare wird **Apertium** in Version 3.2 benötigt. [5]. Es ist angeraten, die aktuellsten Pakete manuell zu kompilieren und dem System hinzuzufügen; zuletzt habe ich folgende Umgebung auf meinem System erfolgreich getestet:

- **Apertium** 3.2 und davon abhängig **lttoolbox** 3.2
- **HFST** 3.3.11 und davon abhängig **OpenFST** 1.3.2 sowie **Foma** 0.9.16alpha
- **VISL CG-3** 0.9.7.8357

Das meiste Kopfzerbrechen mag **HFST** bereiten: Die Nutzung mit **Apertium** erfolgreich zu konfigurieren, erfordert sowohl das "HFST: README" auf dem Wiki des Projekts [6] als auch das **Apertium**-Wiki zum Thema **HFST** [7] zu studieren.

Die anderen Pakete (**OpenFST**, **Foma** und **VISL GC-3**) sind - den Instruktionen in den jeweiligen `README`-Dateien befolgend - leicht installiert.

Wie **Apertium** selber kompiliert werden kann und beliebige (auch experimentelle) Sprachpaare installiert werden können, wird ebenfalls im Wiki des Projekts [8] genau erläutert und funktioniert in aller Regel problemlos. Im Falle, dass ein Sprachpaar gerade nicht kompilierbar ist, helfen die **Apertium**-Entwickler im *IRC*-Chatkanal [9] `#apertium` auf Freenode [10] meinen Erfahrungen gemäss gerne und unkompliziert weiter.

---

3. Dt. "Ich mag es Schokolade zu essen."
4. Dt. "Sie mögen es Schokolade zu essen."
5. Die jüngste für **Debian** verfügbare Fassung, die vorkompiliert paketiert ist, stellt Version 3.1 dar.
6. HFST-Wiki: `https://kitwiki.csc.fi/twiki/bin/view/KitWiki/HfstReadme` (letzter Zugriff: 08.06.2012)
7. **Apertium**-Wiki: `http://wiki.apertium.org/wiki/Hfst` (letzter Zugriff: 08.06.2012)
8. Installation von **Apertium** aus dem **SVN**-Repository: `http://wiki.apertium.org/wiki/Apertium_on_Ubuntu#Installing_the_newest_version_from_SVN_.28more_complicated.29` (letzter Zugriff: 14.06.2012)
9. *Internet Relay Chat*
10. `irc://irc.freenode.net#apertium`

2 Grundlagen: Zu Apertium und HFST

2.1 Wesen und Entwicklungsstand von Apertium

An der ersten **FreeRBMT**-Konferenz[1] rekapitulieren Forcada et al. [3] 2009 unter dem Titel "The Apertium machine translation plattform: five years on" die Entwicklung hin zu einem regelbasierten Übersetzungssystem, das schon damals über 20 funktionale Übersetzungsrichtungen aufweist.

Was aber war fünf Jahre vorher, und was ist jetzt - insgesamt acht Jahre später?

Apertium als solches wurde 2004 als Projekt - finanziert von einem Konsortium aus Staat und Wirtschaft - gegründet, um zunächst die Sprachpaare *es*↔*ca* (Spanisch und Katalanisch) sowie Spanisch und Galicisch als das Paar *es*↔*gl* in beidseitige Übersetzungsrichtungen zu realisieren. Angesiedelt wurde das Projekt zur Umsetzung an der Universität von Alicante. [2]

Zu diesem Zeitpunkt existieren bereits zwei regelbasierte MÜ-Systeme, die als Transfersysteme relativ oberflächlich, ohne tiefe Syntaxanalyse, zwischen ähnlichen Sprachpaaren zu übersetzen fähig sind: **interNOSTRUM** und **Tradutor Universia**. Dieser Ansatz wird als *Shallow-Transfer* bezeichnet und bedeutet im einfachsten Fall, dass ausser einem lexikalischen Transfer kaum weitere Arbeit für das System anfällt eine Übersetzung relativ erfolgreich zu vollziehen. An diesem Ansatz hat sich im Grundsatz nichts geändert. Und: Dieser Ansatz, in seiner einfachsten Transferform, funktioniert nur für sehr ähnliche Sprachpaare gut. Das letztere System (**Tradutor Universia**) ist spezialisiert auf das

1. Eine Konferenz für freie *RBMT*-Systeme: `http://xixona.dlsi.ua.es/freerbmt09/` (letzter Zugriff: 14.06.2012)
2. Webseite der Universität: `http://www.ua.es/` (letzter Zugriff: 14.06.2012)

Sprachpaar *es↔pt*, also Spanisch zu Portugiesisch (und umgekehrt) - als solches ebenso ein Sprachpaar, das eine geringe linguistische Distanz aufweist. Die grössten Unterschiede sind lexikalischer Art.

Vor insgesamt acht Jahren wurde angefangen die zwei o. g. Systeme zu vereinigen. Im Zuge dieser Reimplementation wurde auch der Quellcode unter einer freien Lizenz im Sinne der **Open Source Initiative** [3] veröffentlicht. Jede Person hat somit die Möglichkeit an der Entwicklung von **Apertium** mitzuwirken. Forcada selber war damals leitend mit dabei und mischt heute weiterhin zentral mit. Es sind mittlerweile weitere wichtige Namen in der **Apertium**-Entwicklung involviert, die als Mentoren [4] für Interessierte an der Mitarbeit am Projekt fungieren. Deren unmittelbare und zeitweise ständige Erreichbarkeit kann ich aus eigener Erfahrung bestätigen.

Forcada erwähnt [3], dass für die Sprachpaare *es↔ca* und *es-gl* die sprachtechnologischen Ressourcen teils selber (in seinem akademischen Umfeld) erstellt, oder - wo möglich - aus frei verfügbaren anderen Ressourcen, die frei lizenziert sind, bezogen wurden, um **Apertium** initial aufzubauen.

Angefangen mit Version 1.0, welche für ähnliche Sprachpaare, wie *es↔ca* und *es-pt* ausgelegt ist, bietet Version 2.0 bereits die Möglichkeit zwischen linguistisch weiter entfernten Sprachen zu übersetzen. Als Beispiele hierbei werden *fr↔ca* [5] oder *en↔ca* [6] genannt. Dafür wurde der Transferprozess ausgebaut, denn es genügt bei diesen Sprachen nicht mehr (in vereinfachter Darstellung) die Wörter zu ersetzen. Ein mehrstufiger Strukturtransfer wird erforderlich, ohne aber (bis heute) eine vollständige Syntaxanalyse zu betreiben. Der jüngste Major-Versionssprung zur Version 3 markiert, dass **Apertium** vollständig **UNICODE**-fähig geworden ist und mit allen im Zeichensatz enthaltenen Symbolen operieren kann.

Technologisch setzt **Apertium** auf bewährte Technologien. Die linguistischen Daten, sowohl die Lexika als auch die Transferregeln, werden je Sprachpaar in *XML*-Dateien nach einem klaren Schema abgelegt [7]. Das erhöht die Interoperabilität zu anderen Systemen, die sich diesen Ressourcen bedienen möchten. Ein eigens entwickeltes Toolset existiert (**lttoolbox**), das die *XML*-Dateien in ein binäres Format übersetzen kann, das als ein Finite-State-Transduktor (*FST*) [8] realisiert wird. Aus den Lexika und Transferregeln werden für jedes Sprachpaar letzten Endes drei Transduktoren erstellt, welche folgenden drei Bereichen gewidmet sind, die im Grundsatz den Übersetzungsprozess von **Apertium** ausmachen:

3. Webseite der Initiative: `http://www.opensource.org/` (letzter Zugriff: 14.06.2012)
4. **Apertium**-Wiki: `http://wiki.apertium.org/wiki/List_of_Apertium_mentors` (letzter Zugriff: 14.06.2012)
5. Französisch zu Katalanisch und umgekehrt
6. Englisch zu Katalanisch und umgekehrt
7. Beispiel der Schemata anhand des Wörterbuchschemas: `http://wiki.apertium.org/wiki/Monodix_basics` (letzter Zugriff: 14.06.2012)
8. In diesem konkreten Anwendungsfall genauer bekannt als: *augmented letter transducer*

1. Lexikalische Analyse (der Quellsprachenwörter)

2. Lexikalischer Transfer (anhand der Wortlemmata mit lexikalischen Wortmarkierungen)

3. Lexikalische Generierung (der Zielsprachenwörter)

Um die Lemmata für den Transferprozess erfolgreich lexikalisch zu markieren sind natürlich Zwischenschritte notwendig, wie die *morphologische Analyse* der vorgefundenen Wortform, *PoS-Tagging* [9] zur *Wortartenbestimmung* und *Word Sense Disambiguation*, um das (kontextuell) adäquate Lemma auszuwählen und zu taggen. Je nach Sprachpaar und Sprachbau der involvierten Sprachen ist der dafür erforderliche Aufwand unterschiedlich gross.

Der Einsatz von **HFST**, welcher zur morphologischen Verarbeitung in dieser Hinsicht schwieriger Sprachen genutzt wird, findet im extensivsten Einsatz bei der lexikalischen Analyse (1.) und Generierung statt (2.) - und das auch nur, wenn beide Sprachen der Betrachtung **HFST** zur morphologischen Bewältigung erforderlich machen. Ist nur die Quellsprache morphologisch mit den **lttoolbox**-Werkzeugen *nicht* zu bewältigen, so findet da der Einsatz von **HFST** statt, ansonsten bei der Zielsprache. In Fällen, wo sowohl die Zielsprache als auch die Quellsprache grösstenteils mit *Konkatenativer Morphologie* [10] bewältigt werden kann, macht der Einsatz von **HFST** wenig Sinn. Da wird dann in aller Regel **lttoolbox** für alles verwendet. Das trifft etwa auf alle ursprünglichen Sprachpaare zu, wie ich sie weiter oben erwähnt habe.

Apertium ist der UNIX-Philosophie [11] verschrieben, dass Programme im Einzelnen klein und simpel zu sein haben und erst miteinander in Interaktion befindlich Komplexes schaffen sollen. Durch das verwendete *Pipelining* kann jeder Analyse-, Transfer- und Generierungsschritt in seinem In- und Output genau beobachtet werden. Das und die freie Verfügbarkeit des Quellcodes hilft beim Verständnis der Prozesse ungemein und macht wissenschaftliche Ergebnisse nachvollziehbar. Genauer kann die Pipeline anhand der Abbildung 2.1 illustriert werden. [12]

Die (relative) Unabhängigkeit der einzelnen Komponenten erlaubt es **Apertium** nur in Teilen zu verwenden, z. B. zum *PoS-Tagging* eigener Texte, ungeachtet der weitergehenden und (eigentlichen) Hauptfunktionalität der *Maschinellen Übersetzung*. Somit ist **Apertium** nicht einfach

9. Das PoS-Tagging wird generell statistisch mit trainierten Daten im Einsatz von *Hidden-Markov-Modellen* (*HMM*) gemacht, oder in Fällen, wo zu wenig Datenmaterial vorliegt *oder* der Statistik zugunsten der Linguistik weniger Spielraum gelassen werden soll, fällt auf, dass auch der Constraint Grammar-Formalismus (CG) eingesetzt wird. Das ist beim Sprachpaar *sme-nob*, das in Kapitel 3.2 betrachtetet wird, der Fall.

10. Dazu in 3.1 Genaueres.

11. Vgl. hierzu Wikipedia: `https://en.wikipedia.org/wiki/Unix_philosophy` (letzter Zugriff: 14.06.2012)

12. Die schematische Darstellung entspringt den **TeX**-Quellen der **Apertium**-Dokumentation für Version 3.0, die in Arbeit ist. URL: `http://apertium.svn.sourceforge.net/viewvc/apertium/branches/apertium-documentation/apertium-3.0/en/documentation.tex?view=log&pathrev=38833` (letzter Zugriff: 14.06.2012)

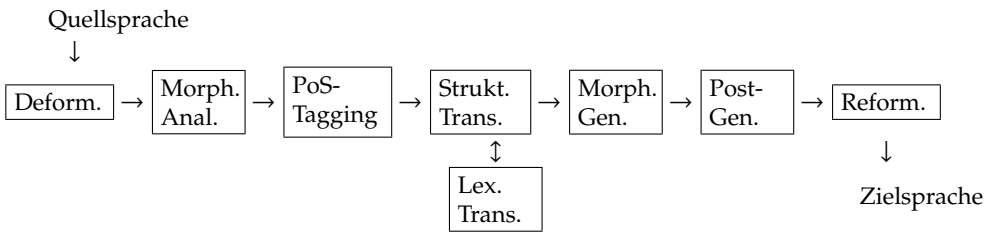

Abbildung 2.1: Es sind acht (abstrakte) Komponenten, die das *Shallow-Transfer*-System **Apertium** ausmachen.

ein abgeschlossenes *MÜ-System*, im Sinne einer Blackbox, sondern eine offene Plattform, die hilft freie sprachtechnologische Ressourcen zu mehren.

Apertium führt die Sprachpaare im **SVN**-Repository [13], abhängig vom Entwicklungsstand, in verschiedenen Ordnern:

- In `incubator/` befinden sich kürzlich angesetzte Sprachpaare.
- In `nursery/` befinden sich noch wenig funktionale Sprachpaare.
- In `staging/` befinden sich bereits fortgeschrittene Sprachpaare.
- In `trunk/` befinden sich stabile Sprachpaare, die produktiv genutzt werden können.

Mittlerweile sind im Vergleich zu 2004 zahlreiche weitere Sprachpaare hinzukommen, entstanden durch Forschungs-/Entwicklungsprojekte von Studierenden [14], Arbeiten von Forschungsgruppen an anderen Forschungseinrichtungen, als Community-Projekte seitens Interessierter der *Freien Software*-Szene und auch mit Entwicklungsunterstützung von Unternehmen mit kommerziellen Interessen, z. B. **Prompsit**, welche Dienstleistungen um **Apertium** anbietet. [15]

Zum Zeitpunkt des Abschlusses dieser Arbeit Mitte Juni 2012[16] befinden sich 138 (!) Sprachpaare in *incubator/* [17], 28 Sprachpaare in *nursery/*, 5 Sprachpaare in *staging/* und 37 Sprachpaare in *trunk/*, womit rund 40 Sprachpaare produktiv genutzt werden können. Forschungs- und Entwicklungsarbeiten hingegen sind schon heute in rund 200 Sprachpaaren möglich und weitere können nach Absprache mit den Hauptentwicklern beliebig hinzugefügt werden.

13. **SVN**-Webview des Repositories: `http://apertium.svn.sourceforge.net/viewvc/` (letzter Zugriff: 14.06.2012)
14. Z. B. im Rahmen des Google Summer of Code (GSoC): `http://wiki.apertium.org/wiki/Google_Summer_of_Code#Active_projects` (letzter Zugriff: 14.06.2012)
15. Prompsit-Webseite: `http://www.prompsit.com/somos-prompsit/` (letzter Zugriff: 14.06.2012)
16. **SVN**-Repository des Projekts in Revision 38833: `http://apertium.svn.sourceforge.net/viewvc/apertium/?pathrev=38833` (letzter Zugriff: 14.06.2012)
17. Sowie einige Morphologien im Ansatz, die keinem konkreten Sprachpaar gewidmet sind.

Wichtig für einen Start sind die freie Verfügbarkeit von Wörterbüchern und gegebenenfalls Morphologien für die involvierten Sprachen. Eine Liste solcher Ressourcen für viele Sprachen wird im **Apertium**-Wiki geführt. [18]

Durch das Hinzukommen von Sprachpaaren, deren Sprachen linguistisch wenig gemein haben, ist der *Shallow-Transfer* inzwischen stärker ausgebaut. Für *en-ca* (Englisch zu Katalanisch) wird ein dreistufiger Strukturtransfer angewandt, in der Reihenfolge *Chunking*, *Inter-* und *Post-Chunking*. Dabei ist zu beachten, dass das *Inter-Chunking* in (noch) komplexeren Fällen mehrstufig sein kann. So ist mir das Sprachpaar *sme-smj* (vom Nordsamischen 3.2 in das Lulesamische [19]) aufgefallen [20], das über drei *Inter-Chunking*-Dateien verfügt [21], die Regeln darüber enthalten *Chunks* (in drei Phasen) zu vertauschen.

2.2 Wesen und Entwicklungsstand von HFST

Beim Helsinki Finite-State Transducer - Framework (**HFST**) handelt es sich um eine in C++ geschriebene Open Source-Softwarekollektion, aus welche die Nutzung verschiedener FST-Technologien und -Formalismen heraus einheitlich möglich ist, die ihrerseits frei (implementiert) verfügbar sind. Es sind dies zum aktuellen Zeitpunkt:

- Die Stuttgart Finite-State Transducer Tools (**SFST**); mit einer Basis-Unterstützung von Finite-State Transducer mit gewichteten Kantenübergängen. [22] [23]
- Das **Foma**-Paket, das eine freie Implementation der **XFST**/**LEXC**-Formalismen darstellt; gewichtete Transduktoren können nicht erstellt werden. [24]
- Die **OpenFST**-Bibliothek; es wird die Schaffung von Finite-State Transducer mit gewichteten Kantenübergängen unterstützt. [25]

Insbesondere bietet **HFST** das Tool *hfst-twolc* an, das einen Two-Level Rule Compiler für den **Xerox-TWOLC**-Formalismus darstellt. Damit können Transduktoren erstellt werden, die eine *parallele* Ersetzung sowohl der Ober- als auch der Unterseite des Transduktors *gleichzeitig* erlauben. Die Reihenfolge der entsprechenden Deklarationen ist damit

18. Apertium-Wiki: `http://wiki.apertium.org/wiki/Specific_resources_per_language` (letzter Zugriff: 14.06.2012)
19. **Ethnologue**-Eintrag für *smj*: `https://www.ethnologue.com/show_language.asp?code=smj` (letzter Zugriff: 14.06.2012)
20. **SVN**-Repository: `http://apertium.svn.sourceforge.net/viewvc/apertium/nursery/apertium-sme-smj/` (letzter Zugriff: 14.06.2012)
21. Die Dateien mit den Endungen ".t2x", ".t3x" und ".t4x" - besonderes Augenmerk kann auf den Bereich ab `<section-rules>` gelegt werden.
22. Webseite: `http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/SFST.html` (letzter Zugriff: 14.06.2012)
23. Gemäss Angaben von Francis M. Tyers im *IRC* wird auf diese Tools / den entsprechenden Bibliotheken in **Apertium** kaum zugegriffen.
24. Dateien mit Endungen ".xfst", ".foma" und ".lexc" werden in aller Regel damit verarbeitet.
25. Dateien mit Endungen ".twol" werden damit prozessiert, wobei von gewichteten Kantenübergängen kein Gebrauch gemacht wird.

unwichtig. Im Gegensatz dazu muss beim (klassischen) **Xerox** Finite-State Tools-Formalismus (**XFST**), welcher **Foma** als Compiler frei implementiert, der Reihenfolge der Deklarationen Beachtung geschenkt werden, weil es sich beim **XFST**-Formalismus um *Rewriting*-Regeln handelt, die sequentiell immer nur auf die Unterseite (bzw. dem "zweiten" oder "rechten" Band) des Transduktors angewandt werden.

Eine umfassende Einführung von Finite-State-Automaten und -Transduktoren würde den Rahmen dieser Arbeit sprengen, es sei allerdings darauf hingewiesen, dass im **Apertium**-Wiki[26] das Thema im Zusammenhang mit den morphologischen Wörterbücher, die mittels den eigenen **lttoolbox**-Werkzeugen erstellt werden, ausreichend eingeführt wird.

Auf der Seite wird ebenfalls plausibel erklärt, dass es zur Übersichtlichkeit und dem Verständnis der Morphologie beiträgt, die Deklaration derselben von den Algorithmen zu trennen, die sie umsetzen. Sowohl der **lttoolbox**-Formalismus als auch die Formalismen, die durch das **HFST**-Framework unterstützt werden, erfordern die blosse Deklaration der Morphologie, ohne dass man sich um die (effiziente) Verarbeitung der Ausdrücke selber kümmern muss. Die prozeduralen Schritte erledigen **lttoolbox** und **HFST** bei der Verarbeitung der jeweiligen Deklarationen in den Dateien [27] selbstständig. Damit wird eine Trennung von morphologischer Deklaration und Programmcode erreicht.

Das ist ähnlich wie bei der deklarativen (logischen) Programmiersprache **Prolog**, wo sich die Inferenzmaschine eigenständig um die effiziente Beweisführung des Ziels (engl. des "goals") kümmert. Das Gegenteil dieses Ansatzes wird auf der Webseite mit **Python**-Code demonstriert, welcher im gegebenen Beispiel dafür verwendet wird, das Wort "beer" und "beers" morphologisch zu analysieren - als ein Substantiv, das im Singular oder Plural steht. Das wird durch prozedurale Schritte direkt erledigt. Bei der Implementation sprachvollständiger Morphologien auf diese Art gestaltet sich die Wartung der Morphologie als tendenziell unübersichtlich. Hinzu können sich bei einem solchen Ansatz in schlechter Implementation Laufzeitprobleme gesellen.
Der wichtigste Vorteil aber bleibt bis hierhin unerwähnt: Transduktoren können nicht nur zur Analyse von Wortformen in ihre Lemmata samt lexikalischer Markierungen genutzt werden [28], sondern markierte Lemmata (im gegebenen Format) können zur Generierung von Wortformen genutzt werden [29] In einer prozeduralen Implementation müsste sowohl Programmcode für die Generierung als auch (separat) für die Analyse geschrieben werden. Das ist umständlich.

---

26. Vgl. das **Apertium**-Wiki: `http://wiki.apertium.org/wiki/Morphological_dictionaries` (letzter Zugriff: 14.06.2012)
27. Es handelt sich für Dateien, die durch **lttoolbox** prozessiert werden, um die Dateien in den Ordner der Sprachpaare mit den Endungen ".dix" für das morphologische Wörterbuch, ".t1x" bis (maximal) ".t5x" (je nach Anzahl der Stufen) für die Transferregeln und gegebenenfalls den ".lexc"- und ".twol"-Dateien, falls bei einer involvierten Sprache die Morphologie durch **HFST** erfasst wird. In wenigen Fällen sind auch ".xfst"- oder ".foma"-Dateien anzutreffen.
28. Der Transduktor wird in einem solchen Fall in der Regel von links nach rechts bzw. von oben nach unten angewandt.
29. Der Transduktor wird von rechts nach links bzw. von unten nach oben angewandt

Ein praktisches Beispiel der Nutzung der resultierenden Transduktoren zur Analyse oder Generierung liefert das **Apertium**-Wiki auf der Seite über **lttoolbox**. [30]

Falls bei einer speziellen Anwendung mit **Apertium** eine (im Einzelfall) weitergehende prozedurale Verarbeitung der Ausgabe aus den Transduktoren notwendig wird, dann besteht die Möglichkeit **HFST** aus z. B. **Python** heraus zu nutzen. [31]

30. Vgl. **Apertium**-Wiki: `http://wiki.apertium.org/wiki/Lttoolbox` (letzter Zugriff: 14.06.2012)
31. Vgl. Beitrag auf der *nltk-dev*-Mailingliste: `https://groups.google.com/group/nltk-dev/browse_thread/thread/fdb53482cad56234` (letzter Zugriff: 14.06.2012)

3 Hauptteil: Die Integration von HFST in Apertium

3.1 Theorie: Zweck und Nutzen der Integration

Finite-State-Technologien können in der Computerlinguistik dafür genutzt werden, um Wörterbücher elegant aufzubauen oder die morphologische Analyse und Generierung von Wörtern systematisch zu betreiben *und* das insbesondere auch in Fällen, wo die Morphologie der betrachteten Sprache nicht relativ einfach durch z. B. Prä- oder Suffixe (also konkatenativ durch 'Aneinanderreihen' von Morphemen) erfasst werden kann, sondern gerade auch in Fällen, wo auch andere Affixe, wie Infixe, zur Anwendung kommen.

Nicht mehr simpel sind Morphologien beispielsweise in Fällen, wo es für gegebene Wortstämme [1] Fälle gibt, wo Morpheme für eine Flexion inmitten des Stamms eingepflanzt werden müssen oder wo Vokale inmitten von Stämmen wegfallen oder durch andere ersetzt werden müssen, um gültige Wortformen analysieren oder generieren zu können. Morphologien für Sprachen, die komplexe Anpassungen an Stämmen oder fortgesetzten Wortklassen in bestimmten (z. B. grammatikalischen) Fällen erfordern, werden auch *Nicht-Konkatenative Morphologien* (NKM) (vgl. Clematide 2007: 104ff.) genannt, oder zumindest wäre die Konkatenation nach vorwiegend erkennbaren systematischen Regeln der betrachteten Sprache nicht mehr (sinnvoll) nachvollziehbar, würde man eine *Konkatenative Morphologie* erzwingen wollen; ebenfalls ist die Gefahr von Redundanz sehr hoch, wenn jeder nur denkbare Stamm in eine Morphologie einfliesst.

Eine saubere Morphologie mit klaren Regeln und in (möglichst) schlanker Ausführung ist eine Anforderung, von der gerade ein *Maschinelles Übersetzungssystem* wie **Apertium** fundamental profitiert, um schnell zu

1. Minimaloberfläche von Wörtern, aus der durch Morpheme, die vor-, um - oder angehängt werden können, weitere wichtige (weitere) Wortformen entstehen.

arbeiten und (linguistisch nachvollziehbar) gepflegt werden zu können.

Wie bereits in Kapitel 2.1 angedeutet, ist es nicht erst das **HFST**-Framework, das Finite-State-Technologien in **Apertium** einführt. Die lexikalischen Werkzeuge **lttoolbox** mit Anwendungen wie *lt-comp* [2] oder *lt-proc* [3] werden dazu genutzt die in *XML* abgefassten Wörterbücher in Transduktoren umzuwandeln - für eine schnellere Verarbeitung. Mit den Möglichkeiten verglichen, die das **HFST**-Framework liefert, sind die **lttoolbox**-Werkezuge am ehesten mit dem **LEXC**-Formalismus vergleichbar, welcher **Foma** beherrscht und auf die Idee fusst Wörterbücher anhand von Stämmen (von z. B. Verben oder Nomen) anhand von Fortsetzungsklassen (durch das Anhängen von Morphemen) aufzubauen. Simon Clematide führt auch das in seinem Vorlesungsskript von 2007 (93ff.) genauer aus. Darin ist es möglich die *Morphotaktik* der Sprache zu erfassen. Schwieriger umzusetzen mit diesen **Apertium**-Bordmitteln ist die *Morphophonemik*, worunter *Lautanpassungen* oder die *Infigierung* [4] fallen. Diese (u. U. seltenen, aber für die Korrektheit notwendigen) Anpassungen der *Morphotaktik* berechtigen den Beizug zusätzlicher (wohl bekannter und anerkannter) Formalismen, wie **XFST** [5] oder **TWOLC**.

Die wenigsten natürlichen Sprachen sind (vollständig) *Konkatenativer Morphologie*. Im Rahmen einer kürzlichen Veranstaltung zu *Finite-State-Methoden* (bei Simon Clematide) haben Simon Hafner und ich die Möglichkeit erhalten eine (konzeptionelle) Morphologie [6] für die Plansprache "Klingonisch" [7] anzulegen. Diese lässt sich tatsächlich fast vollständig mit Fortsetzungsklassen auf Basis der Wortstämme und anzuhängenden Morphemen und damit dem **LEXC**-Formalismus erfassen, ist aber auch nur ein akademisches Beispiel einer Sprache, wo es nie zu (natürlichen) Lautverschiebungen oder anderen komplexen Phänomenen kam.

Die Dokumentation darüber, wie **HFST** in **Apertium** genau genutzt wird, ist spärlich. Gerade in der **Apertium**-Dokumentation (aktuell nachgeführt und veröffentlicht bis Version 2.0) seitens Forcada et al. [2] wird nichts über die Nutzung von **HFST** ausgesagt - auch nicht in den neuesten (versionskontrollierten) Fassungen[8] im **SVN**-Repository.

Glücklicherweise waren wichtige **Apertium**-Entwickler mir eine grosse Hilfe dabei die nötigen Einstiegspunkte zu finden. Ein Beispiel einer typischen Hilfeleistung ist im Anhang als "IRC-Konversation zu HFST" [9] zu finden, die ich mit freundlicher Unterstützung der involvierten Personen in dieser Seminararbeit in der Form publizieren darf.

---

2. Zur Kompilation von Wörterbüchern von einem *XML*- in ein FST-Format
3. Zur Prozessierung der Daten im vorgängig generierten FST-Format und der Ausgabe in einem menschenlesbaren lexikalischen Tagging-Format
4. Einfügen eines Affixes in einen Wortstamm, auch Infix genannt
5. Von **Foma** implementiert
6. Code und Anwendungsbeispiele bei **github** im **git**-Repository: `https://github.com/2mh/klingomorph` (letzter Zugriff: 14.06.2012)
7. **Ethnologue**-Eintrag *tlh*: `https://www.ethnologue.com/show_language.asp?code=tlh` (letzter Zugriff: 14.06.2012)
8. Vgl. hierzu: `http://apertium.svn.sourceforge.net/viewvc/apertium/trunk/apertium-documentation/apertium-2.0/en/` (letzter Zugriff: 12.06.2012)
9. Siehe Anhang auf Seite 26.

Zudem: Es sind im **Apertium**-Wiki Informationen darüber vorhanden, wie ein neues Sprachpaar für die Nutzung mit **HFST** angelegt werden kann. [10] Beim gegebenen Beispiel wird schematisch das Sprachpaar *tur-tuk* angesetzt (real im **SVN**-Repository existiert das Sprachpaar *tuk-tur* [11]), wobei *tuk* [12]für Turkmenisch und *tur* [13] für Türkisch steht.

Generell wird bei Sprachpaaren, die mit **HFST** analysiert (und generiert) werden, versucht den lexikalischen Transfer möglichst auf Basis der Morpheme vorzunehmen. Idealerweise kann der grösste Teil im **LEXC**-Formalismus abgedeckt werden. Wo das nicht (im Ansatz) gelingt, werden **TWOLC**-Regeln definiert, welche die nötigen Anpassungen an den lexikalischen Markierungen *und* Oberflächenformen der Wörter vornehmen.

Gründe, weshalb **TWOLC**- gegenüber **XFST**-Deklarationen bevorzugt werden, sind gemäss wichtigen **Apertium**-Entwicklern [14] (1) das übersichtlichere Format und (2) die einfachere Handhabung von linguistischen Phänomenen wie der *Vokalharmonie* mit dem **TWOLC**-Formalismus.

Ein weiterer wichtiger Grund, welcher die mangelnde Verbreitung des **XFST**-Formalismus bei **Apertium** erklärt, ist vermutlich auch, dass **Foma** erst seit Version 3.0 von **HFST** (veröffentlicht in 2011) vollständig unterstützt wird.

3.2 Empirie: Die Integration am Beispiel eines Sprachpaars

Darüber wie ein neues Sprachpaar unter Nutzung von **HFST** zu **Apertium** hinzugefügt werden kann, gibt ein Wiki-Artikel [15] Auskunft. Im Sinne eines Tutorials wird dabei Schritt für Schritt im Ansatz gezeigt, wie die Implementation einer *RBMT* von der Turkmenischen Sprache ins Türkische (Sprachpaar: *tk-tr*) umgesetzt werden kann. Bei beiden verwandten Sprachen handelt es sich um stark agglutinierende Sprachen.

Das (wissenschaftlich) am besten dokumentierte Beispiel der Nutzung von **HFST** im Zusammenhang mit **Apertium** wird im noch nicht publizierten Paper "Evaluating North Sámi to Norwegian assimilation RBMT" [4] beschrieben, das von Trond Trosterud und seinem Kollegen

10. Vgl. hierzu: `http://wiki.apertium.org/wiki/Starting_a_new_language_with_HFST` (letzter Zugriff: 12.06.2012)
11. Vgl. hierzu: `http://apertium.svn.sourceforge.net/viewvc/apertium/nursery/apertium-tuk-tur/` (letzter Zugriff: 12.06.2012)
12. **Ethnologue**-Eintrag für *tuk*: `https://www.ethnologue.com/show_language.asp?code=tuk` (letzter Zugriff: 12.06.2012)
13. **Ethnologue**-Eintrag für *tur*: `https://www.ethnologue.com/show_language.asp?code=tur` (letzter Zugriff: 12.06.2012)
14. Gemäss Francis M. Tyers und Jonathan North Washington im *IRC*
15. Apertium-Wiki: `http://wiki.apertium.org/wiki/Starting_a_new_language_with_HFST` (letzter Zugriff: 11.06.2012)

Kevin Brubeck Unhammer Mitte Juni 2012 an der **FreeRBMT** 2012 erstmals öffentlich vorgetragen wird. [16]

Die Nordsamische Sprache [17] (*sme*) verfügt über geschätzte 15'000 bis 25'000 Sprecher und wird in Norwegen, Schweden und Finnland gesprochen. Die Sprache wird von den Autoren als stark flektierend und agglutinierend eingeteilt.

Die Standardvarietät [18] des Norwegischen, in die übersetzt wird, verfügt gemäss **Ethnologue** [19] über rund 4.5 Millionen Sprecher und gehört der indogermanischen Sprachfamilie an. Die Sprache wird von Trosterud und Unhammer [4] als morphologisch wenig komplex angegeben.

Dieses Sprachpaar gehört zu den wenigen, wo eine indoeuropäische (oder -germanische) Sprache unter Einsatz von **HFST** mit **Apertium** übersetzt wird.

Bei den meisten anderen Sprachpaaren, wo **HFST** genutzt wird, handelt es sich beidseitig um Sprachen mit komplexer Morphologie.

Bezüglich des Sprachpaars unserer Betrachtung fokussieren die Autoren auf die Übersetzungsrichtung ins Norwegische, weil der Anspruch bloss ist, dass Nur-Norwegischsprechende Texte aus dem Nordsamischen verstehen können sollen. Umgekehrt würden die meisten Sprecher des Nordsamischen bereits Norwegisch verstehen, womit der Nutzen der umgekehrten Übersetzungsrichtung als (zum aktuellen Zeitpunkt) gering eingeschätzt wird.

Als Motivation für den Einsatz von **HFST** gegenüber **lttoolbox** wird beispielhaft der *Stufenwechsel* der Konsonanten des Nordsamischen, in *Quantität* als auch *Qualität*, angeführt. [20] Generalisiert betrachtet, befürworten die Autoren die Verwendung von **HFST**, wenn Morphologien zu verarbeiten sind, die nicht rein konkatenativ sind - wie ich in Kapitel 3.1 zum *Zweck* von **HFST** bereits ausgeführt habe.

Grundsätzlich wird gemäss Entwickler Unhammer [21] (schematisch) die folgende UNIX-Pipeline für *sme-nob* angewandt:

```
$ echo words|hfst-proc sme-analyser.hfst|...| \\ #1
lt-proc sme-to-nob-dix.bin|...| \\ #2
lt-proc nob-generator-dix.bin #3
```

Im Falle von *sme-nob* wird **HFST** alleine für die Analyse (#1) von *sme* genutzt, dann wird wieder generell auf **lttoolbox** gesetzt, in wichtigster Hinsicht im Rahmen des *lexikalischen Transfers* (#2) und der *Generierung*

16. Information gemäss einem Gespräch mit Kevin Brubeck Unhammer im *IRC*; Konferenzseite mit Programm: `http://www.molto-project.eu/freerbmt-program.html` (letzter Zugriff: 14.06.2012)
17. Engl. "North Sámi"
18. Als Buchsprache "Bokmål"
19. **Ethnologue**-Eintrag zu den Sprachen Norwegens: `https://www.ethnologue.com/show_country.asp?name=no` (letzter Zugriff: 11.06.2012)
20. Vgl. hierzu Wikipedia: `https://de.wikipedia.org/wiki/Stufenwechsel` (letzter Zugriff: 14.06.2012)
21. Chatgespräch im *IRC*

(#3) der Zielsprache *nob*.

Der morphologische *sme*-Analyzer ist dabei in den Formalismen **LEXC** und **TWOLC** geschrieben und fusst auf bereits verfügbare Ressourcen [22] zurück, was aufzeigt, dass **HFST** im Einsatz für **Apertium** auch deshalb interessant ist, weil viele Morphologien bereits existieren, die in Formalismen geschrieben sind, die **HFST** verarbeiten kann. Für das Anlegen vieler Sprachpaare ist es nicht notwendig alles *from scratch* zu beginnen.

Auch beim *nob*-Generator konnte auf (eigene frühere) Ressourcen zurückgegriffen werden, wie diese im Sprachpaar *nn-nb* [23] verfügbar sind; bei dem Sprachpaar wird zwischen zwei norwegischen Sprachverietäten übersetzt. Wichtig ist noch einmal zu betonen, dass der *nob*-Teil (zur Generation der Norwegischen Wortformen) auf **lttoolbox** setzt und mit **HFST** nichts zu schaffen hat.

Interessant ist der Einsatz der von Trosterud und Unhammer von Grund auf selber geschriebene *Constraint Grammar* (*CG*) [24] zur *Wortartendisambiguierung*, in dem die Auswahl der Wörter (für die Zielsprache) anhand von (lokalen) grammatikalischen Kontexten (in der Quellsprache) geschieht. Das ist auch der Grund, weshalb ich eingangs in Kapitel 1.2 **VISL CG-3** als obligat zu installierendes Paket anführe. Ohne diese Software ist es nicht möglich das Sprachpaar *sme-nob* zu kompilieren. [25]

Zwischen der *morphologischen Analyse* (von *sme*) und der *morphologischen Generierung* von *nob* findet der *strukturelle Transfer* statt, der 4-phasig ist; die Autoren Trosterud und Unhammer machen hierbei folgende Angaben:

1. 63 *Chunk*-Regeln (`t1x`-Datei)
2. 26 *Interchunk1*-Regeln (`t2x`-Datei)
3. 39 *Interchunk2*-Regeln (`t3x`-Datei)
4. 29 *Postchunk*-Regeln (`t4x`-Datei)

Es reicht also nicht bloss aus Wortformen oder Morpheme zu ersetzen, sondern es sind komplexe Verschiebungen der markierten Lemmata

22. **SVN**-Repository des Giellatekno-Projekts: `https://victorio.uit.no/langtech/trunk/gt/sme` (letzter Zugriff: 14.06.2012)
23. **SVN**-Repository: `http://apertium.svn.sourceforge.net/viewvc/apertium/trunk/apertium-nn-nb/` (letzter Zugriff: 14.06.2012)
24. Vgl. für einen kurzen Überblick eine CLab-Seite: `http://kitt.cl.uzh.ch/clab/constraintGrammar/ilap_visl/` (letzter Zugriff: 14.06.2012)
25. Zu erwähnen ist in dem Zusammenhang, dass bei den meisten Sprachpaaren *Hidden-Markov-Modelle* zum Einsatz kommen, um die Wortarten zu bestimmen; dafür wird Sprachmaterial trainiert. Im Falle von *sme-nob* deuten die Autoren allerdings darauf hin, dass mangels repräsentativen Sprachmaterials diese Option keine ist, die befriedigen kann. Sie stellen allerdings in Aussicht, dass eine Kombination von linguistischen *CG*- und statistischen *HMM*-Ansätzen ein Versuch wert wäre. Dabei würde zuerst mittels *CG* ein Kontext eingegrenzt und danach statistisch verfahren.

notwendig, um in das *Norwegische* zu übersetzen, obschon beide Sprachen der *Subjekt-Verb-Objekt*-Satzstellung sind.

Die Evaluationsergebnisse der Autoren deuten darauf hin, dass das Sprachpaar *sme-nob* funktionalen Charakter hat. *Post-Editing*-Qualität ist nicht das Ziel und wird auch nicht erreicht, allerdings können Norwegischsprechende, die kein Nordsamisch verstehen, in vielen Fällen ausmachen, was die Bedeutung eines Satzes ist - insbesondere dann, wenn er kontextualisiert vorliegt.

Im Folgenden soll eine praktische Nutzung des Sprachpaars *sme-nob* in **Apertium** illustriert werden. Als Beispiel diene der einleitende Begrüssungssatz zur Nordsamischen Wikipedia [26] ausgewählt:

```
$ echo 'Bures boahtin Wikipediai, friddja diehtosátnegirjái!'
\\ | apertium sme-nob
Velkommen til Wikipedia, til den frie informasjonsordboken!
```

Die resultierende *nob*-Übersetzung lautet gemäss Google Translate [27] auf Deutsch:

> Willkommen bei Wikipedia, dem freien Wörterbuch Informationen!

Die Übersetzung ist verständlich, abgesehen von der Semantik des Ausdrucks "Wörterbuch Informationen". Hier sollte meiner Ansicht nach "Enzyklopädie" stehen. Es ist wahrscheinlich, dass bereits die Quelle nicht den optimalen Begriff führt und da das eigentliche Problem liegt.

Interessanter ist, was passiert, wenn wir das ominöse *i*-Suffix bei "Wikipediai" im Nordsamischen entfernen:

```
$ echo 'Bures boahtin Wikipedia, friddja diehtosátnegirjái!'
\\ | apertium sme-nob
Velkommen Wikipedia, til den frie informasjonsordboken!
```

Wir können feststellen, dass der Satz einen anderen personalen Bezug herstellt, denn gemäss Google Translate [28] bedeutet die zweite *nob*-Übersetzung auf Deutsch:

> Willkommen Wikipedia, dem freien Wörterbuch Informationen!

Wir werden nun also als "Wikipedia" angesprochen.

---

26. Webseite: `https://se.wikipedia.org/wiki/V%C3%A1ldosiidu` (letzter Zugriff: 14.06.2012)
27. Vgl. hierzu: `http://translate.google.com/#no|de|Velkommen%20til%20Wikipedia%2C%20til%20den%20frie%20informasjonsordboken!` (letzter Zugriff: 14.06.2012)
28. Vgl. hierzu: `http://translate.google.com/#no|de|%0AVelkommen%20Wikipedia%2C%20til%20den%20frie%20informasjonsordboken\mskip-\thinmuskip` (letzter Zugriff: 14.06.2012)

Eine *morphologische Analyse* der beiden Wortformen "Wikipedia" und "Wikipediai" erhärtet die Vermutungen, dass es sich beim *i*-Suffix um einen *Illativ* [29] handelt, einem Lokalkasus, der dazu genutzt werden kann eine "Hineinbewegung" auszudrücken, respektive in diesem Fall die entsprechende Präposition "auf" in das Substantiv / der *Named Entity* "Wikipedia" einzugliedern:

```
$ echo "Wikipedia" | apertium -d . sme-nob-morph
#1:
^Wikipedia/Wikipedia<N><Prop><Org><Sg><Acc>/
Wikipedia<N><Prop><Org><Sg><Gen>/
Wikipedia<N><Prop><Org><Sg><Nom>$^./.<CLB>$

#2:
$ echo "Wikipediai" | apertium -d . sme-nob-morph
^Wikipediai/Wikipedia<N><Prop><Org><Sg><Ill>$^./.<CLB>$
```

Bei Analyse #1 wird "Wikipedia" als ein Substantiv , eine *Named Entity* und (semantisch) als Organisation im Singular markiert. Uneinigkeit herrscht für den Tagger auf dieser Transferstufe noch über den Kasus: Es ist nicht klar, ob "Wikipedia" im Nominativ, Akkusativ oder Genitiv steht. Das Symbol <CLB> steht für "Clause Boundary" und spielt insbesondere für den *Chunker* (im Zuge des späteren Transfers) eine Rolle, z. B. zur (groben) Abgrenzung von Nominal- und Verbalphrasen.

Analyse #2 fällt knapper aus, da der Kasusfall keine Rolle mehr spielt - an dessen Stelle tritt nun der Illativ (Symbol <Ill>). Der Rest bleibt sich gleich.

Da aber die Illativ-Markierung in diesem spezifischen Fall auf der *nob*-Seite zur Oberflächengenerierung des norwegischen Wortes "til" [30] führt, ändert sich die Semantik des Satzes merklich.

Der Transferprozess bei **Apertium** kann generell beliebig fein beobachtet werden; für unser spezifisches Sprachpaar *sme-nob* sei für weitere Versuche auf das **Apertium**-Wiki [31] verwiesen, das extensiv Informationen darüber führt.

Es existieren weitere interessante Sprachpaare, bei der nur eine Seite jeweils stark agglutierend ist und auf entsprechende Formalismen (**LEXC**- und **TWOLC**-Formalismen) zurückgegriffen wird: So etwa die

29. Vgl. Wikipedia: https://de.wikipedia.org/wiki/Illativ (letzter Zugriff: 14.06.2012)
30. Dt. "auf"
31. http://wiki.apertium.org/wiki/Northern_S%C3%A1mi_and_Norwegian (letzter Zugriff: 14.06.2012)

Sprachpaare *quz-spa* (Quecha-Castellano) [32], *udm-rus* (Udmuritische Sprache-Russisch) [33] und *kaz-eng* (Kasachisch-Englisch). [34] Noch ist der Entwicklungsstand all dieser Übersetzungsrichtungen aber gering und für einen produktiven Einsatz oder eine Evaluation demnach ungeeignet. [35]

32. **Apertium**-Wikiseite: `http://wiki.apertium.org/wiki/Quechua_cuzque%C3%B1o_y_castellano` (letzter Zugriff: 14.06.2012)
33. Im **SVN**-Repository: `http://apertium.svn.sourceforge.net/viewvc/apertium/nursery/apertium-udm-rus/` (letzter Zugriff: 14.06.2012)
34. Im **SVN**-Repository: `http://apertium.svn.sourceforge.net/viewvc/apertium/incubator/apertium-eng-kaz/` (letzter Zugriff: 14.06.2012)
35. Alle drei Sprachpaare sind in den **SVN**-Ordnern `incubator/` oder `nursery/` untergebracht.

4 Schluss

4.1 Zusammenfassung und Ausblick

In abschliessender Betrachtung ist festzuhalten: Die Nutzung von **HFST** verleiht **Apertium** die Fähigkeit zu und von Sprachen mit (stark) *Nicht-Konkatenativer Morphologie* zu übersetzen. Dafür müssen die **Apertium**-eigenen **lttoolbox**-Werkzeuge umgangen werden, die ansonsten zur *morphologischen Analyse* und *morphologischen Generierung* von Wortformen verwendet werden. Der Einsatz von **HFST** liefert aber keinen Mehrwert an anderen Stellen in der Transfer-Pipeline. Der *lexikalische Transfer* oder strukturelle Anpassungen, um der Syntax der Zielsprache näher zu kommen, sind Schritte, für die sich die **lttoolbox**-Werkzeuge weiterhin bewähren.

Diskutiert wird eine stärkere Integration von **HFST** in **Apertium**, damit **HFST** nicht länger der Charakter eines "Fremdkörpers" in der **Apertium**-Plattform anhaftet. Es existiert die Idee ein Projekt in diese Richtung anzustossen, um es z. B. möglich zu machen `.lexc`-Dateien anstatt mit den **HFST**-Binaries mit den **lttoolbox**-Kommandos zu prozessieren. Dieses Projekt könnte von einem interessierten Studierenden etwa im Rahmen des nächsten Google Summer of Code (2013) initiiert werden. [1]

Ein weitere Herausforderung, die in der Natur von **Apertium** als *Shallow-Transfer-System* begründet liegt [2], hat mit den Bilexika zu tun. **Apertium** muss für jedes Sprachpaar ein Bilexikon führen, um den *lexikalischen Transfer* mittels den **lttoolbox**-Werkzeugen zu vollziehen. Da für jede neue Sprachkombination das Bilexikon neu aufgebaut werden muss, besteht hier ein gewisser (manueller) Aufwand bei jedem neuen Sprachpaar, das angesetzt wird; dieser kann zwar durch Einsatz von Werkzeuge aus den **apertium-dixtools** minimiert werden, die Vollautomation

1. Idee zum Projekt: `http://wiki.apertium.org/wiki/Ideas_for_Google_Summer_of_Code/Closer_integration_with_HFST` (letzter Zugriff: 14.06.2012)
2. **Apertium** verwendet keinen sog. *Interlingua*-Ansatz um eine sprachunabhängige Repräsentation von Wortbedeutungen zu führen.

funktioniert aber noch nicht. [3] [4]

Weiterhin bleibt **Apertium** im Grundsatz ein *Shallow-Transfer*-System, das keinen vollständigen Parse der Quellsprache vornimmt und sie tiefen-analysiert. Dadurch ist es auch in Zukunft nicht breit möglich Semantik über weite Distanzen zu transferieren.

Bezüglich weiterer Möglichkeiten für **Apertium** morphologisch zu analysieren / generieren, kann das Augenmerk auch auf neuere Formalismen als jene von **Xerox** [5] gelegt werden. So existiert **hunmorph** [6], das es erlaubt in *OCaml* [7] - in einem objekt-orientierten Paradigma - Morphologien zu beschreiben und dazu eine (relativ) einfache Syntax aufzuweisen scheint. Beispiele explorativer Versuche mit **hunmorph** für **Apertium** sind im Wiki des Projekts bereits heute dokumentiert. [8] Ausserdem ist **hunmoph** morphologisch-analytische Basis des bekannten *Spellcheckers* **HunSpell**, das der standardmässige *Spellchecker* in **LibreOffice**, **OpenOffice.org** und einigen Softwareprodukten der **Mozilla Foundation** ist. [9]

Selber hat es mir Freude bereitet mich in dieses komplexe Gebiet hinein zu begeben und ich bin für die Zukunft interessiert daran mich an **Apertium** nach Kräften zu beteiligen, da ich die dahinterstehende Community als eine kennen gelernt habe, die aufgeschlossen, humorvoll und im höchsten Masse kompetent ist.

4.2 Danksagungen

Ich möchte mich an dieser Stelle herzlichst im Besonderen bei folgenden Personen des **Apertium**-Projekts bedanken, dass sie sich Zeit genommen haben sich mit mir im *IRC* ausgiebig zu unterhalten und den Dschungel in und um **Apertium** zu durchleuchten: Mikel L. Forcada, Francis M. Tyers, Kevin Brubeck Unhammer und Jonathan North Washington.

Ohne sie wäre diese Arbeit in der Form nicht entstanden und vieles Quellmaterial, das ich hier verwende, schwierg auffindbar gewesen.

3. **SVN**-Repository: `http://apertium.svn.sourceforge.net/viewvc/apertium/trunk/apertium-dixtools/` (letzter Zugriff: 14.06.2012)
4. Vgl. hierzu: `http://wiki.apertium.org/wiki/Crossdics` (letzter Zugriff: 14.06.2012)
5. Namentlich **XFST**, **TWOLC** und **LEXC**
6. Webseite: `http://mokk.bme.hu/resources/hunmorph` (letzter Zugriff: 14.06.2012)
7. Webseite: `http://caml.inria.fr/` (letzter Zugriff: 14.06.2012)
8. **Apertium**-Wiki: `http://wiki.apertium.org/wiki/Hunmorph` (letzter Zugriff: 14.06.2012)
9. Webseite: `http://hunspell.sourceforge.net/` (letzter Zugriff: 14.06.2012)

Literaturverzeichnis

[1] S. Clematide. Morphologie und Lexikographie. Vorlesungsskript. Institut für Computerlinguistik, Universität Zürich, 2007. URL http://files.ifi.uzh.ch/cl/siclemat/lehre/ss07/mul/script/script.pdf.

[2] M.L. Forcada, B.I. Bonev, S. Ortiz-Rojas, J.A. Pérez-Ortiz, G. Ramírez-Sánchez, F. Sánchez-Martínez, C. Armentano-Oller, M.A. Montava, F.M. Tyers und U. dA. Departament de Llenguatges i Sistemes Informàtics. Documentation of the open-source shallow-transfer machine translation platform apertium. 2010. URL http://xixona.dlsi.ua.es/~fran/apertium2-documentation.pdf.

[3] M.L. Forcada, F.M. Tyers und G. Ramírez-Sánchez. The Apertium machine translation platform: five years on. In *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation*, 3–10. 2009. URL http://xixona.dlsi.ua.es/freerbmt09/presentations/forcada_freerbmt09_5yrson.pdf.

[4] T. Trosterud und K.B. Unhammer. Evaluating North Sámi to Norwegian assimilation RBMT. In *Proceedings of the Third International Workshop on Free/Open-Source Rule-Based Machine Translation*. 2012. URL http://apertium.svn.sourceforge.net/viewvc/apertium/trunk/apertium-sme-nob/paper/?pathrev=38833. (noch kein offizielles Paper (im PDF) verfügbar).

Abbildungsverzeichnis

Anhang

IRC-Konversation zu HFST

Typisches Beispiel einer Konversation [10] mit Jonathan North Washington ("firespeaker") und Mikel L. Forcada ("mlforcada"), welche wesentliche Beiträge zu **Apertium** leisten; geführt im *IRC* [11]:

```
19:21 < h2m> At which stage of the pipeline is HFST employed?
19:21 < h2m> I couldn't get that
19:21 < mlforcada> h2m generation of Kazakh
[...]
19:22 < h2m> Is this process described somewhere in detail,
such that I can read that up?
19:22 < mlforcada> is something that looks like lttoolbox but
more powerful
19:22 < mlforcada> h2m, firespeaker can help probably
19:22 < h2m> I can only see *.lexc and *.twol files w/o really
getting the point.
[...]
19:22 < firespeaker> h2m: HFST takes a string of text and turns
it into lemmas + tags
19:22 < firespeaker> h2m: or
19:22 < firespeaker> h2m: it takes lemmas + tags and turns it
into a string of text
[...]
19:23 < firespeaker> h2m: .lexc is the morphotactics
19:23 < mlforcada> firespeaker, you're right
19:23 < h2m> Ah, OK
19:23 < firespeaker> h2m: .twol is the morphophonology
```

10. Selber bin ich als "h2m" unterwegs.
11. `irc://irc.freenode.net#apertium`