Universität Zürich[UZH]
Institut für Computerlinguistik
Dozenten: Dr. Cerstin Mahlow, Dr.-Ing. Michael Piotrowski

Sprachvarianten des Deutschen im Zeitraum 1600 bis heute

Vorlesung: Sprachtechnologie für historische Dokumente: Konzepte und Anwendungen (HS 2011)

Simon Hafner, Hernani Marques Madeira, Reto Baumgartner

16. Dezember 2011

Problemstellung

Erkennung der Entstehungszeit
historischer Texte nach 1600

- Im Rahmen des Projekts OLdPhras[1]
- Unser Code (public domain) verfügbar. URL:
  `https://github.com/2mh/OLdPhras-langvar`
- Lösungsansatz: Mit Hilfe eines n-Gramm-Sprachmodells (statistisch)
- Frage: Wo liegen die besten Resultate – bei welcher n-Gramm-Ordnung?

[1] URL: `http://colloc.germa.unibas.ch/oldphras/` (17. 12. 2011)

Trainingskorpus (Reto Baumgartner)

Daten für das Trainingskorpus

Deutsches Textarchiv[2]

Bereich Literatur

Nach den Regeln der Text Encoding Initiative (TEI-P5[3])

[2] URL: http://www.deutschestextarchiv.de/ (17.12.2011)

[3] URL: http://www.tei-c.org/index.xml (17.12.2011)

TEI-Format

Mehrere Werke in einem XML-File

Einzelne Werke repräsentiert durch Knoten mit dem Tag «TEI»

- Literaturgenre:
  - `TEI/teiHeader/profileDesc/textClass/keywords/term`
  - `prose`, `drama`, `verse`
- Erstellungsdatum:
  - `TEI/teiHeader/profileDesc/creation/date`
  - `notBefore="1840" notAfter="1902"` (z. B. bei Texten von Emile Zola, geb. 1840, † 1902)
- Text:
  - `TEI/text`

TEI-Format

Mehrere Werke in einem XML-File

Einzelne Werke repräsentiert durch Knoten mit dem Tag «TEI»

- Literaturgenre:
  - `TEI/teiHeader/profileDesc/textClass/keywords/term`
  - `prose`, `drama`, `verse`
- Erstellungsdatum:
  - `TEI/teiHeader/profileDesc/creation/date`
  - `notBefore="1840" notAfter="1902"` (z. B. bei Texten von Emile Zola, geb. 1840, † 1902)
- Text:
  - `TEI/text`

TEI-Format

Mehrere Werke in einem XML-File

Einzelne Werke repräsentiert durch Knoten mit dem Tag «TEI»

- Literaturgenre:
  - `TEI/teiHeader/profileDesc/textClass/keywords/term`
  - `prose`, `drama`, `verse`
- Erstellungsdatum:
  - `TEI/teiHeader/profileDesc/creation/date`
  - `notBefore="1840" notAfter="1902"` (z. B. bei Texten von Emile Zola, geb. 1840, † 1902)
- Text:
  - `TEI/text`

Werkzeug zur Erstellung des Trainingskorpus

`korpusbastler.py`

In Python codiert

Werkzeug zur Erstellung des Trainingskorpus

Für jedes enthaltene Werk:

- Literaturgenre:
  - Weiterverarbeitung der Genres `prose`, `drama`
  - Keine Textextraktion bei Genres wie `verse`
- Erstellungsdatum:
  - `notBefore, notAfter`: Lebensdaten des Autors
  - Mögliches Erstellungsjahr: Mitte zwischen den Jahren
  - Einteilung in Korpora nach halben Jahrhunderten mithilfe des Erstellungsjahres
- Text:
  - Extraktion des Textes auf allen Tiefen
  - Mit `xml.etree.cElementTree.itertext()`
  - Schreiben in entsprechende Korpusdateien

Werkzeug zur Erstellung des Trainingskorpus

Für jedes enthaltene Werk:

- Literaturgenre:
  - Weiterverarbeitung der Genres `prose`, `drama`
  - Keine Textextraktion bei Genres wie `verse`
- Erstellungsdatum:
  - `notBefore, notAfter`: Lebensdaten des Autors
  - Mögliches Erstellungsjahr: Mitte zwischen den Jahren
  - Einteilung in Korpora nach halben Jahrhunderten mithilfe des Erstellungsjahres
- Text:
  - Extraktion des Textes auf allen Tiefen
  - Mit `xml.etree.cElementTree.itertext()`
  - Schreiben in entsprechende Korpusdateien

Werkzeug zur Erstellung des Trainingskorpus

Für jedes enthaltene Werk:

- Literaturgenre:
  - Weiterverarbeitung der Genres `prose`, `drama`
  - Keine Textextraktion bei Genres wie `verse`
- Erstellungsdatum:
  - `notBefore, notAfter`: Lebensdaten des Autors
  - Mögliches Erstellungsjahr: Mitte zwischen den Jahren
  - Einteilung in Korpora nach halben Jahrhunderten mithilfe des Erstellungsjahres
- Text:
  - Extraktion des Textes auf allen Tiefen
  - Mit `xml.etree.cElementTree.itertext()`
  - Schreiben in entsprechende Korpusdateien

Trainingskorpora für die Sprachmodelle

Sprachstufe	Anzahl Wörter im Korpus	
1600–1650	2’574’487	(rund 16 MB)
1650–1700	5’652’759	(rund 34 MB)
1700–1750	2’809’613	(rund 18 MB)
1750–1800	9’101’048	(rund 56 MB)
1800–1850	19’325’579	(rund 118 MB)
1850–1900	27’389’361	(rund 169 MB)
1900–	2’396’123	(rund 15 MB)

Testkorpus

Testkorpora

- Aus `http://de.wikisource.org/`
- 100 Sätze pro Sprachstufe
- d. h. 20 Sätze pro Jahrzehnt
- Ausnahme bei Testsatz ab 1900 (rund 10 Sätze je Jahrzehnt)
- Genres entsprechen dem Trainingskorpus

Testkorpora

- Aus `http://de.wikisource.org/`
- 100 Sätze pro Sprachstufe
- d. h. 20 Sätze pro Jahrzehnt
- Ausnahme bei Testsatz ab 1900 (rund 10 Sätze je Jahrzehnt)
- Genres entsprechen dem Trainingskorpus

Testkorpora

- Aus `http://de.wikisource.org/`
- 100 Sätze pro Sprachstufe
- d. h. 20 Sätze pro Jahrzehnt
- Ausnahme bei Testsatz ab 1900 (rund 10 Sätze je Jahrzehnt)
- Genres entsprechen dem Trainingskorpus

Testkorpora

- Aus `http://de.wikisource.org/`
- 100 Sätze pro Sprachstufe
- d. h. 20 Sätze pro Jahrzehnt
- Ausnahme bei Testsatz ab 1900 (rund 10 Sätze je Jahrzehnt)
- Genres entsprechen dem Trainingskorpus

Testkorpora

- Aus `http://de.wikisource.org/`
- 100 Sätze pro Sprachstufe
- d. h. 20 Sätze pro Jahrzehnt
- Ausnahme bei Testsatz ab 1900 (rund 10 Sätze je Jahrzehnt)
- Genres entsprechen dem Trainingskorpus

Sprachmodelle (Hernani Marques)

n-Gramm-Wahrscheinlichkeiten (bedingt)

- Basierend auf Übungen in PCL3-HS11
- Zeichenbasierte n-Gramm-Modelle: 1- bis 6-Gramme
- Somit: Insg. 12 Modelle
- Smoothing für unvorhandene Sequenzen (arbiträr): $10^{(} - 8)$

n-Gramm-Wahrscheinlichkeiten (bedingt)

- Basierend auf Übungen in PCL3-HS11
- Zeichenbasierte n-Gramm-Modelle: 1- bis 6-Gramme
- Somit: Insg. 12 Modelle
- Smoothing für unvorhandene Sequenzen (arbiträr): $10^{(} - 8)$

n-Gramm-Wahrscheinlichkeiten (bedingt)

- Basierend auf Übungen in PCL3-HS11
- Zeichenbasierte n-Gramm-Modelle: 1- bis 6-Gramme
- Somit: Insg. 12 Modelle
- Smoothing für unvorhandene Sequenzen (arbiträr): $10^{(} - 8)$

n-Gramm-Wahrscheinlichkeiten (bedingt)

- Basierend auf Übungen in PCL3-HS11
- Zeichenbasierte n-Gramm-Modelle: 1- bis 6-Gramme
- Somit: Insg. 12 Modelle
- Smoothing für unvorhandene Sequenzen (arbiträr): $10^{(} - 8)$

Werkzeug zum Training der Daten

`langMod.py`

- Funktion *generate_ngrams()* liefert n-Gramme zurück (zeichenbasiert); ist ein Generator
- Klasse *LM* speichert (bedingte) n-Gramm-Wahrscheinlichkeiten pro Sprachvariante
- Klasse *MLM* speichert LM-Modelle

Werkzeug zum Training der Daten

`langMod.py`

- Funktion *generate_ngrams()* liefert n-Gramme zurück (zeichenbasiert); ist ein Generator
- Klasse *LM* speichert (bedingte) n-Gramm-Wahrscheinlichkeiten pro Sprachvariante
- Klasse *MLM* speichert LM-Modelle

Training der Daten

- `timeRanges = [ "1600_1650", "1650_1700", "1700_1750", "1750_1800", "1800_1850", "1850_1900", "1900_2010"]`
  - Training von sieben (deutschen) Sprachvarianten in 50er Jahre Blöcke; Ausnahme bei modernen Sprache
- `gramOrders = range(1,7) # for n-gram order 1-6`
  - n-Gramm-Ordnungen: 1-6 (5 und 6 haben eine (sehr) kritische Komplexität)

Werkzeug zur Sprachidentifikation

`langMod.py`

- Funktion *accuracy* liefert wahrscheinlichste Sprachvariante für eine Testzeile zurück
  - Ein Objekt der Klasse `MLM` führt wahrscheinlichste Sprache für diese Zeile an
- Anhand 100 Testzeilen (Dateien *e100-**) von Sätzen der jeweiligen Sprachvariante messen wir die Genauigkeit (Anzahl Treffer / 100)

Werkzeug zur Sprachidentifikation

`langMod.py`

- Funktion *accuracy* liefert wahrscheinlichste Sprachvariante für eine Testzeile zurück
  - Ein Objekt der Klasse `MLM` führt wahrscheinlichste Sprache für diese Zeile an
- Anhand 100 Testzeilen (Dateien *e100-**) von Sätzen der jeweiligen Sprachvariante messen wir die Genauigkeit (Anzahl Treffer / 100)

Ergebnisse (Simon Hafner)

Verbesserungsmöglichkeiten (Simon Hafner)

Trainingskorpus

- Menge Sprachmaterial (ungleich verteilt aktuell)
- Textsorten (potenziell ungleich verteilt)
- Nachbearbeitung des Sprachmaterials u. U. nötig

Trainingskorpus

- Menge Sprachmaterial (ungleich verteilt aktuell)
- Textsorten (potenziell ungleich verteilt)
- Nachbearbeitung des Sprachmaterials u. U. nötig

Trainingskorpus

- Menge Sprachmaterial (ungleich verteilt aktuell)
- Textsorten (potenziell ungleich verteilt)
- Nachbearbeitung des Sprachmaterials u. U. nötig

Trainingskorpus

- Menge Sprachmaterial (ungleich verteilt aktuell)
- Textsorten (potenziell ungleich verteilt)
- Nachbearbeitung des Sprachmaterials u. U. nötig

Testsätze

- Auswahl ist (zufällig) auf Wikisource erfolgt
- Sätze vs. Absätze (letztere (= länger) liefern akuratere Resultate)
- Textsorten (potenziell ungleich verteilt, insb. ab 1900 bis heute)
  - Rechtliche Texte dominieren ab 1940er auf Wikisource
  - Andere Textsorten tendenziell nicht public domain

Testsätze

- Auswahl ist (zufällig) auf Wikisource erfolgt
- Sätze vs. Absätze (letztere (= länger) liefern akuratere Resultate)
- Textsorten (potenziell ungleich verteilt, insb. ab 1900 bis heute)
  - Rechtliche Texte dominieren ab 1940er auf Wikisource
  - Andere Textsorten tendenziell nicht public domain

Testsätze

- Auswahl ist (zufällig) auf Wikisource erfolgt
- Sätze vs. Absätze (letztere (= länger) liefern akuratere Resultate)
- Textsorten (potenziell ungleich verteilt, insb. ab 1900 bis heute)
  - Rechtliche Texte dominieren ab 1940er auf Wikisource
  - Andere Textsorten tendenziell nicht public domain

Testsätze

- Auswahl ist (zufällig) auf Wikisource erfolgt
- Sätze vs. Absätze (letztere (= länger) liefern akuratere Resultate)
- Textsorten (potenziell ungleich verteilt, insb. ab 1900 bis heute)
  - Rechtliche Texte dominieren ab 1940er auf Wikisource
  - Andere Textsorten tendenziell nicht public domain

Ergebnisse

	1600-1650	1650-1700	1700-1750	1750-1800	1
Unigramm	0.54	0.5	0.42	0.09	
Bigramm	0.73	0.54	0.31	0.25	
Trigramm	0.65	0.68	0.22	0.46	
4-gramm	0.66	0.8	0.2	0.5	
5-gramm	0.53	0.82	0.12	0.46	
6-gramm	0.39	0.86	0.1	0.32	

Kritikpunkte

- Weitere statistische Masse einführen
- Smoothing-Algorithmus
- Wort-n-Gramme

Kritikpunkte

- Weitere statistische Masse einführen
- Smoothing-Algorithmus
- Wort-n-Gramme

Kritikpunkte

- Weitere statistische Masse einführen
- Smoothing-Algorithmus
- Wort-n-Gramme

Codekomplexität/-Ausführungsgeschwindigkeit

- Das Trainieren von Daten dauert lange (insb. bei Wort-n-Grammen)
- Multiprozessor-Support einbauen
- Algorithmische Verbesserungen (Zeit-/Raumkomplexität)
- Andere Programmiersprache verwenden

Codekomplexität/-Ausführungsgeschwindigkeit

- Das Trainieren von Daten dauert lange (insb. bei Wort-n-Grammen)
- Multiprozessor-Support einbauen
- Algorithmische Verbesserungen (Zeit-/Raumkomplexität)
- Andere Programmiersprache verwenden

Codekomplexität/-Ausführungsgeschwindigkeit

- Das Trainieren von Daten dauert lange (insb. bei Wort-n-Grammen)
- Multiprozessor-Support einbauen
- Algorithmische Verbesserungen (Zeit-/Raumkomplexität)
- Andere Programmiersprache verwenden

Codekomplexität/-Ausführungsgeschwindigkeit

- Das Trainieren von Daten dauert lange (insb. bei Wort-n-Grammen)
- Multiprozessor-Support einbauen
- Algorithmische Verbesserungen (Zeit-/Raumkomplexität)
- Andere Programmiersprache verwenden

Codekomplexität/-Ausführungsgeschwindigkeit

- Das Trainieren von Daten dauert lange (insb. bei Wort-n-Grammen)
- Multiprozessor-Support einbauen
- Algorithmische Verbesserungen (Zeit-/Raumkomplexität)
- Andere Programmiersprache verwenden

Fragen