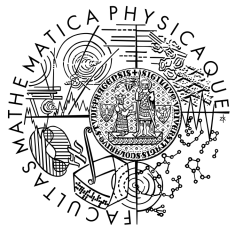
Splitting and Identifying Czech Compounds: A Pilot Study

Emil Svoboda

Charles University, Faculty of Mathematics and Physics



Joint work with Magda Ševčíková for DeriMo 2021

Table of contents

Compounding

Definitions and terminology

"The formation of a new lexeme by adjoining two or more lexemes" (Bauer, 2003)

vždy	+ ***zelený***	→ ***vždyzelený***	
always.ADV	green.ADJ	evergreen.ADJ	

- the two or more input lexemes are "parents" or "parent words"

This paper focuses on:

- Czech, where compounding is underresearched computationally
- graphical words only:
  - strings of graphical symbols unbroken by whitespace or special characters

"The formation of a new lexeme by adjoining two or more lexemes" (Bauer, 2003)

vždy	+	***zelený***	→ ***vždyzelený***
always.ADV		green.ADJ	evergreen.ADJ

- the two or more input lexemes are "parents" or "parent words"

This paper focuses on:

- Czech, where compounding is underresearched computationally
- graphical words only:
  - strings of graphical symbols unbroken by whitespace or special characters

"The formation of a new lexeme by adjoining two or more lexemes" (Bauer, 2003)

vždy	+	***zelený***	→	***vždyzelený***
always.ADV		green.ADJ		evergreen.ADJ

- the two or more input lexemes are "parents" or "parent words"

This paper focuses on:

- Czech, where compounding is underresearched computationally
- graphical words only:
  - strings of graphical symbols unbroken by whitespace or special characters

"The formation of a new lexeme by adjoining two or more lexemes" (Bauer, 2003)

vždy	+	***zelený***	→	***vždyzelený***
always.ADV		green.ADJ		evergreen.ADJ

- the two or more input lexemes are "parents" or "parent words"

This paper focuses on:

- Czech, where compounding is underresearched computationally
- graphical words only:
  - strings of graphical symbols unbroken by whitespace or special characters

"The formation of a new lexeme by adjoining two or more lexemes" (Bauer, 2003)

vždy	+	***zelený***	→	***vždyzelený***
always.ADV		green.ADJ		evergreen.ADJ

- the two or more input lexemes are "parents" or "parent words"

This paper focuses on:

- Czech, where compounding is underresearched computationally
- graphical words only:
  - strings of graphical symbols unbroken by whitespace or special characters

"The formation of a new lexeme by adjoining two or more lexemes" (Bauer, 2003)

vždy	+	***zelený***	→	***vždyzelený***
always.ADV		green.ADJ		evergreen.ADJ

- the two or more input lexemes are "parents" or "parent words"

This paper focuses on:

- Czech, where compounding is underresearched computationally
- graphical words only:
  - strings of graphical symbols unbroken by whitespace or special characters

"The formation of a new lexeme by adjoining two or more lexemes" (Bauer, 2003)

vždy	+	***zelený***	→	***vždyzelený***
always.ADV		green.ADJ		evergreen.ADJ

- the two or more input lexemes are "parents" or "parent words"

This paper focuses on:

- Czech, where compounding is underresearched computationally
- graphical words only:
  - strings of graphical symbols unbroken by whitespace or special characters

"The formation of a new lexeme by adjoining two or more lexemes" (Bauer, 2003)

vždy	+	***zelený***	→	***vždyzelený***
always.ADV		green.ADJ		evergreen.ADJ

- the two or more input lexemes are "parents" or "parent words"

This paper focuses on:

- Czech, where compounding is underresearched computationally
- graphical words only:
  - strings of graphical symbols unbroken by whitespace or special characters

"The formation of a new lexeme by adjoining two or more lexemes" (Bauer, 2003)

vždy	+	***zelený***	→	***vždyzelený***
always.ADV		green.ADJ		evergreen.ADJ

- the two or more input lexemes are "parents" or "parent words"

This paper focuses on:

- Czech, where compounding is underresearched computationally
- graphical words only:
  - strings of graphical symbols unbroken by whitespace or special characters

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený	$\rightarrow$	***vždy***	***zelený***
evergreen.ADJ		always.ADV	green.ADJ

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený	$\rightarrow$	***vždy***	***zelený***
evergreen.ADJ		always.ADV	green.ADJ

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený	$\rightarrow$	***vždy***	***zelený***
evergreen.ADJ		always.ADV	green.ADJ

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený	→	***vždy***	***zelený***
evergreen.ADJ		always.ADV	green.ADJ

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený	→	***vždy***	***zelený***
evergreen.ADJ		always.ADV	green.ADJ

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený	→	***vždy***	***zelený***
evergreen.ADJ		always.ADV	green.ADJ

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený	→	***vždy***	***zelený***
evergreen.ADJ		always.ADV	green.ADJ

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený	→	***vždy***	***zelený***
evergreen.ADJ		always.ADV	green.ADJ

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený	→	*vždy*	*zelený*
evergreen.ADJ		always.ADV	green.ADJ

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený → *vždy* *zelený*
evergreen.ADJ always.ADV green.ADJ

The tasks at hand

Compound identification and compound splitting

Compound identification:

- Binary classification
- Input graphical word **W** (assumed to be a Czech lexeme):
  - if **W** is a Czech compound, return 1;
  - else, return 0.

Compound splitting:

- An open-ended task
- Input graphical word **W** (assumed to be a Czech compound):
  - return *linguistically valid* parent *lemmas* of **W**, separated by spaces.

vždyzelený	→	***vždy***	***zelený***
evergreen.ADJ		always.ADV	green.ADJ

Challenges

What makes the tasks difficult?

krv-o-tok → ***krev*** ***tok***
bloodflow.NOU blood.NOU flow.NOU

modr-o-oký → ***modrý*** ***oko***
blue-eyed.NOU blue.NOU eye.NOU

ps-o-vod → ***pes*** ***vést***
dog handler.NOU dog.NOU lead.VERB

holekchtivý → ***chtivý*** ***holka*** (***holek*** = gen. pl.)
wanting girls.ADJ wanting.ADJ girl.NOU

dvaapůlletý → ***dvě*** ***a*** ***půl*** ***léto***
two-and-a-half-year-old.ADJ two.NUM and.CONJ half.NUM year.ADJ

soci-o-logie → ***-soci-*** ***-log-***
sociology.NOU -soci-.NEOCON -log-.NEOCON

tetrachlorethylen → ***-tetra-*** ***chlor*** ***ethylen***
tetrachlorethylene.NOU -tetra-.NEOCON chlorine.NOU ethylene.NOU

Challenges

What makes the tasks difficult?

krv-o-tok → ***krev*** ***tok***
bloodflow.NOU blood.NOU flow.NOU

modr-o-oký → ***modrý*** ***oko***
blue-eyed.NOU blue.NOU eye.NOU

ps-o-vod → ***pes*** ***vést***
dog handler.NOU dog.NOU lead.VERB

holekchtivý → ***chtivý*** ***holka*** (***holek*** = gen. pl.)
wanting girls.ADJ wanting.ADJ girl.NOU

dvaapůlletý → ***dvě*** ***a*** ***půl*** ***léto***
two-and-a-half-year-old.ADJ two.NUM and.CONJ half.NUM year.ADJ

soci-o-logie → ***-soci-*** ***-log-***
sociology.NOU -soci-.NEOCON -log-.NEOCON

tetrachlorethylen → ***-tetra-*** ***chlor*** ***ethylen***
tetrachlorethylene.NOU -tetra-.NEOCON chlorine.NOU ethylene.NOU

Challenges

What makes the tasks difficult?

krv-o-tok → ***krev*** ***tok***
bloodflow.NOU blood.NOU flow.NOU

modr-o-oký → ***modrý*** ***oko***
blue-eyed.NOU blue.NOU eye.NOU

ps-o-vod → ***pes*** ***vést***
dog handler.NOU dog.NOU lead.VERB

holekchtivý → ***chtivý*** ***holka*** (***holek*** = gen. pl.)
wanting girls.ADJ wanting.ADJ girl.NOU

dvaapůlletý → ***dvě*** ***a*** ***půl*** ***léto***
two-and-a-half-year-old.ADJ two.NUM and.CONJ half.NUM year.ADJ

soci-o-logie → ***-soci-*** ***-log-***
sociology.NOU -soci-.NEOCON -log-.NEOCON

tetrachlorethylen → ***-tetra-*** ***chlor*** ***ethylen***
tetrachlorethylene.NOU -tetra-.NEOCON chlorine.NOU ethylene.NOU

Challenges

What makes the tasks difficult?

krv-o-tok → ***kr̲ev*** ***tok***
bloodflow.NOU blood.NOU flow.NOU

modr-o-oký → ***modrý*** ***oko***
blue-eyed.NOU blue.NOU eye.NOU

ps-o-v̲o̲d → ***pe̲s*** ***vé̲st***
dog handler.NOU dog.NOU lead.VERB

hole̲kchtivý → ***chtivý*** ***holka*** (***holek*** = gen. pl.)
wanting girls.ADJ wanting.ADJ girl.NOU

dvaapůlletý → ***dvě*** ***a*** ***půl*** ***léto***
two-and-a-half-year-old.ADJ two.NUM and.CONJ half.NUM year.ADJ

soci-o-logie → ***-soci-*** ***-log-***
sociology.NOU -soci-.NEOCON -log-.NEOCON

tetrachlorethylen → ***-tetra-*** ***chlor*** ***ethylen***
tetrachlorethylene.NOU -tetra-.NEOCON chlorine.NOU ethylene.NOU

Challenges

What makes the tasks difficult?

krv-o-tok → ***krev*** ***tok***
bloodflow.NOU blood.NOU flow.NOU

modr-o-oký → ***modrý*** ***oko***
blue-eyed.NOU blue.NOU eye.NOU

ps-o-vod → ***pes*** ***vést***
dog handler.NOU dog.NOU lead.VERB

holekchtivý → ***chtivý*** ***holka*** (***holek*** = gen. pl.)
wanting girls.ADJ wanting.ADJ girl.NOU

dvaapůlletý → ***dvě*** ***a*** ***půl*** ***léto***
two-and-a-half-year-old.ADJ two.NUM and.CONJ half.NUM year.ADJ

soci-o-logie → ***-soci-*** ***-log-***
sociology.NOU -soci-.NEOCON -log-.NEOCON

tetrachlorethylen → ***-tetra-*** ***chlor*** ***ethylen***
tetrachlorethylene.NOU -tetra-.NEOCON chlorine.NOU ethylene.NOU

Challenges

What makes the tasks difficult?

krv-o-tok → ***krev*** ***tok***
bloodflow.NOU blood.NOU flow.NOU

modr-o-oký → ***modrý*** ***oko***
blue-eyed.NOU blue.NOU eye.NOU

ps-o-vod → ***pes*** ***vést***
dog handler.NOU dog.NOU lead.VERB

holekchtivý → ***chtivý*** ***holka*** (***holek*** = gen. pl.)
wanting girls.ADJ wanting.ADJ girl.NOU

dvaapůlletý → ***dvě*** ***a*** ***půl*** ***léto***
two-and-a-half-year-old.ADJ two.NUM and.CONJ half.NUM year.ADJ

soci-o-logie → ***-soci-*** ***-log-***
sociology.NOU -soci-.NEOCON -log-.NEOCON

tetrachlorethylen → ***-tetra-*** ***chlor*** ***ethylen***
tetrachlorethylene.NOU -tetra-.NEOCON chlorine.NOU ethylene.NOU

Challenges

What makes the tasks difficult?

krv-o-tok → ***krev*** ***tok***
bloodflow.NOU blood.NOU flow.NOU

modr-o-oký → ***modrý*** ***oko***
blue-eyed.NOU blue.NOU eye.NOU

ps-o-vod → ***pes*** ***vést***
dog handler.NOU dog.NOU lead.VERB

holekchtivý → ***chtivý*** ***holka*** (***holek*** = gen. pl.)
wanting girls.ADJ wanting.ADJ girl.NOU

dvaapůlletý → ***dvě*** ***a*** ***půl*** ***léto***
two-and-a-half-year-old.ADJ two.NUM and.CONJ half.NUM year.ADJ

soci-o-logie → ***-soci-*** ***-log-***
sociology.NOU -soci-.NEOCON -log-.NEOCON

tetrachlorethylen → ***-tetra-*** ***chlor*** ***ethylen***
tetrachlorethylene.NOU -tetra-.NEOCON chlorine.NOU ethylene.NOU

Challenges

What makes the tasks difficult?

krv-o-tok → ***krev*** ***tok***
bloodflow.NOU blood.NOU flow.NOU

modr-o-oký → ***modrý*** ***oko***
blue-eyed.NOU blue.NOU eye.NOU

ps-o-vod → ***pes*** ***vést***
dog handler.NOU dog.NOU lead.VERB

holekchtivý → ***chtivý*** ***holka*** (***holek*** = gen. pl.)
wanting girls.ADJ wanting.ADJ girl.NOU

dvaapůlletý → ***dvě*** ***a*** ***půl*** ***léto***
two-and-a-half-year-old.ADJ two.NUM and.CONJ half.NUM year.ADJ

soci-o-logie → ***-soci-*** ***-log-***
sociology.NOU -soci-.NEOCON -log-.NEOCON

tetrachlorethylen → ***-tetra-*** ***chlor*** ***ethylen***
tetrachlorethylene.NOU -tetra-.NEOCON chlorine.NOU ethylene.NOU

Challenges

What makes the tasks difficult?

krv-o-tok → ***krev*** ***tok***
bloodflow.NOU blood.NOU flow.NOU

modr-o-oký → ***modrý*** ***oko***
blue-eyed.NOU blue.NOU eye.NOU

ps-o-vod → ***pes*** ***vést***
dog handler.NOU dog.NOU lead.VERB

holekchtivý → ***chtivý*** ***holka*** (***holek*** = gen. pl.)
wanting girls.ADJ wanting.ADJ girl.NOU

dvaapůlletý → ***dvě*** ***a*** ***půl*** ***léto***
two-and-a-half-year-old.ADJ two.NUM and.CONJ half.NUM year.ADJ

soci-o-logie → ***-soci-*** ***-log-***
sociology.NOU -soci-.NEOCON -log-.NEOCON

tetrachlorethylen → ***-tetra-*** ***chlor*** ***ethylen***
tetrachlorethylene.NOU -tetra-.NEOCON chlorine.NOU ethylene.NOU

Background

The ways (Czech) compounds have been handled so far

- Theoretical accounts:
  - Bozděchová (1997) uses an onomasiological approach
  - Štichauer (2003) discusses Bisetto and Scalise's (2005) approach wrt Czech
  - Ološtiak and Vojteková (2021) cover neoclassical compounding
- Static data resources:
  - DerivAnCze: contains no compounds
  - DeriNet 2.0: contained 33,923 identified and 1,252 split compounds
- Procedural tools:

Authors	Language	Task	Performance
Krotova et al. (2020)	German	Splitting	Acc: 96%
Hellwig and Nehrdich (2018)	Sanskrit	Splitting	Acc: 96%
Clouet and Daille (2014)	English	Finding split-points	F1: 80%
Clouet and Daille (2014)	Russian	Finding split-points	F1: 63%

Background

The ways (Czech) compounds have been handled so far

- Theoretical accounts:
  - Bozděchová (1997) uses an onomasiological approach
  - Štichauer (2003) discusses Bisetto and Scalise's (2005) approach wrt Czech
  - Ološtiak and Vojteková (2021) cover neoclassical compounding
- Static data resources:
  - DerivAnCze: contains no compounds
  - DeriNet 2.0: contained 33,923 identified and 1,252 split compounds
- Procedural tools:

Authors	Language	Task	Performance
Krotova et al. (2020)	German	Splitting	Acc: 96%
Hellwig and Nehrdich (2018)	Sanskrit	Splitting	Acc: 96%
Clouet and Daille (2014)	English	Finding split-points	F1: 80%
Clouet and Daille (2014)	Russian	Finding split-points	F1: 63%

Background

The ways (Czech) compounds have been handled so far

- Theoretical accounts:
  - Bozděchová (1997) uses an onomasiological approach
  - Štichauer (2003) discusses Bisetto and Scalise's (2005) approach wrt Czech
  - Ološtiak and Vojteková (2021) cover neoclassical compounding
- Static data resources:
  - DerivAnCze: contains no compounds
  - DeriNet 2.0: contained 33,923 identified and 1,252 split compounds
- Procedural tools:

Authors	Language	Task	Performance
Krotova et al. (2020)	German	Splitting	Acc: 96%
Hellwig and Nehrdich (2018)	Sanskrit	Splitting	Acc: 96%
Clouet and Daille (2014)	English	Finding split-points	F1: 80%
Clouet and Daille (2014)	Russian	Finding split-points	F1: 63%

Background

The ways (Czech) compounds have been handled so far

- Theoretical accounts:
  - Bozděchová (1997) uses an onomasiological approach
  - Štichauer (2003) discusses Bisetto and Scalise's (2005) approach wrt Czech
  - Ološtiak and Vojteková (2021) cover neoclassical compounding
- Static data resources:
  - DerivAnCze: contains no compounds
  - DeriNet 2.0: contained 33,923 identified and 1,252 split compounds
- Procedural tools:

Authors	Language	Task	Performance
Krotova et al. (2020)	German	Splitting	Acc: 96%
Hellwig and Nehrdich (2018)	Sanskrit	Splitting	Acc: 96%
Clouet and Daille (2014)	English	Finding split-points	F1: 80%
Clouet and Daille (2014)	Russian	Finding split-points	F1: 63%

Background

The ways (Czech) compounds have been handled so far

- Theoretical accounts:
  - Bozděchová (1997) uses an onomasiological approach
  - Štichauer (2003) discusses Bisetto and Scalise's (2005) approach wrt Czech
  - Ološtiak and Vojteková (2021) cover neoclassical compounding
- Static data resources:
  - DerivAnCze: contains no compounds
  - DeriNet 2.0: contained 33,923 identified and 1,252 split compounds
- Procedural tools:

Authors	Language	Task	Performance
Krotova et al. (2020)	German	Splitting	Acc: 96%
Hellwig and Nehrdich (2018)	Sanskrit	Splitting	Acc: 96%
Clouet and Daille (2014)	English	Finding split-points	F1: 80%
Clouet and Daille (2014)	Russian	Finding split-points	F1: 63%

Background

The ways (Czech) compounds have been handled so far

- Theoretical accounts:
  - Bozděchová (1997) uses an onomasiological approach
  - Štichauer (2003) discusses Bisetto and Scalise's (2005) approach wrt Czech
  - Ološtiak and Vojteková (2021) cover neoclassical compounding
- Static data resources:
  - DerivAnCze: contains no compounds
  - DeriNet 2.0: contained 33,923 identified and 1,252 split compounds
- Procedural tools:

Authors	Language	Task	Performance
Krotova et al. (2020)	German	Splitting	Acc: 96%
Hellwig and Nehrdich (2018)	Sanskrit	Splitting	Acc: 96%
Clouet and Daille (2014)	English	Finding split-points	F1: 80%
Clouet and Daille (2014)	Russian	Finding split-points	F1: 63%

Background

The ways (Czech) compounds have been handled so far

- Theoretical accounts:
  - Bozděchová (1997) uses an onomasiological approach
  - Štichauer (2003) discusses Bisetto and Scalise's (2005) approach wrt Czech
  - Ološtiak and Vojteková (2021) cover neoclassical compounding
- Static data resources:
  - DerivAnCze: contains no compounds
  - DeriNet 2.0: contained 33,923 identified and 1,252 split compounds
- Procedural tools:

Authors	Language	Task	Performance
Krotova et al. (2020)	German	Splitting	Acc: 96%
Hellwig and Nehrdich (2018)	Sanskrit	Splitting	Acc: 96%
Clouet and Daille (2014)	English	Finding split-points	F1: 80%
Clouet and Daille (2014)	Russian	Finding split-points	F1: 63%

Background

The ways (Czech) compounds have been handled so far

- Theoretical accounts:
  - Bozděchová (1997) uses an onomasiological approach
  - Štichauer (2003) discusses Bisetto and Scalise's (2005) approach wrt Czech
  - Ološtiak and Vojteková (2021) cover neoclassical compounding
- Static data resources:
  - DerivAnCze: contains no compounds
  - DeriNet 2.0: contained 33,923 identified and 1,252 split compounds
- Procedural tools:

Authors	Language	Task	Performance
Krotova et al. (2020)	German	Splitting	Acc: 96%
Hellwig and Nehrdich (2018)	Sanskrit	Splitting	Acc: 96%
Clouet and Daille (2014)	English	Finding split-points	F1: 80%
Clouet and Daille (2014)	Russian	Finding split-points	F1: 63%

Background

The ways (Czech) compounds have been handled so far

- Theoretical accounts:
  - Bozděchová (1997) uses an onomasiological approach
  - Štichauer (2003) discusses Bisetto and Scalise's (2005) approach wrt Czech
  - Ološtiak and Vojteková (2021) cover neoclassical compounding
- Static data resources:
  - DerivAnCze: contains no compounds
  - DeriNet 2.0: contained 33,923 identified and 1,252 split compounds
- Procedural tools:

Authors	**Language**	**Task**	**Performance**
Krotova et al. (2020)	German	Splitting	Acc: 96%
Hellwig and Nehrdich (2018)	Sanskrit	Splitting	Acc: 96%
Clouet and Daille (2014)	English	Finding split-points	F1: 80%
Clouet and Daille (2014)	Russian	Finding split-points	F1: 63%

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

ryb-o-lov → ***ryba*** ***lov***
fishery.NOU fish.NOU hunt.NOU

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

Adjective 1 + *-o-* + Adjective 2 → Compound Adjective
důležitý + ***-o-*** + ***neomylný*** → ***důležitoneomylný***
important.ADJ + -o- + infallible.ADJ → *important-infallible.ADJ

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

ryb-o-lov → ***ryba*** ***lov***
fishery.NOU fish.NOU hunt.NOU

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

Adjective 1 + *-o-* + Adjective 2 → Compound Adjective
důležitý **+ *-o-* +** ***neomylný*** **→ *důležitoneomylný***
important.ADJ + -o- + infallible.ADJ → *important-infallible.ADJ

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

$$\underset{\text{fishery.NOU}}{\textit{ryb-o-lov}} \rightarrow \underset{\text{fish.NOU}}{\textit{ryba}} \quad \underset{\text{hunt.NOU}}{\textit{lov}}$$

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

$$\textbf{Adjective 1} + \textit{-o-} + \textbf{Adjective 2} \rightarrow \textbf{Compound Adjective}$$
$$\textit{důležitý} + \textit{-o-} + \textit{neomylný} \rightarrow \textit{důležitoneomylný}$$
$$\text{important.ADJ} + \text{-o-} + \text{infallible.ADJ} \rightarrow \text{*important-infallible.ADJ}$$

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

ryb-o-lov → ***ryba*** ***lov***
fishery.NOU fish.NOU hunt.NOU

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

Adjective 1 + *-o-* + Adjective 2 → Compound Adjective
důležitý + ***-o-*** + ***neomylný*** → ***důležitoneomylný***
important.ADJ + -o- + infallible.ADJ → *important-infallible.ADJ

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

ryb-o-lov → ***ryba*** ***lov***
fishery.NOU fish.NOU hunt.NOU

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

Adjective 1 + *-o-* + Adjective 2 → Compound Adjective
důležitý + ***-o-*** + ***neomylný*** → ***důležitoneomylný***
important.ADJ + -o- + infallible.ADJ → *important-infallible.ADJ

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

ryb-o-lov → ***ryba*** ***lov***
fishery.NOU fish.NOU hunt.NOU

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

Adjective 1 + *-o-* + Adjective 2 → Compound Adjective
důležitý + ***-o-*** + ***neomylný*** → ***důležitoneomylný***
important.ADJ + -o- + infallible.ADJ → *important-infallible.ADJ

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

ryb-o-lov	→	***ryba***	***lov***
fishery.NOU		fish.NOU	hunt.NOU

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

Adjective 1	+	***-o-*** +	**Adjective 2**	→ **Compound Adjective**
důležitý		+ ***-o-*** +	***neomylný***	→ ***důležitoneomylný***
important.ADJ		+ -o- +	infallible.ADJ	→ *important-infallible.ADJ

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

ryb-o-lov → ***ryba*** ***lov***
fishery.NOU fish.NOU hunt.NOU

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

Adjective 1 + *-o-* + Adjective 2 → Compound Adjective
důležitý + *-o-* + ***neomylný*** → ***důležitoneomylný***
important.ADJ + -o- + infallible.ADJ → *important-infallible.ADJ

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

ryb-o-lov → ***ryba*** ***lov***
fishery.NOU fish.NOU hunt.NOU

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

Adjective 1 + *-o-* + Adjective 2 → Compound Adjective
důležitý + ***-o-*** + ***neomylný*** → ***důležitoneomylný***
important.ADJ + -o- + infallible.ADJ → *important-infallible.ADJ

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

ryb-o-lov → ***ryba*** ***lov***
fishery.NOU fish.NOU hunt.NOU

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

Adjective 1 + -*o*- + Adjective 2 → Compound Adjective
důležitý + -*o*- + ***neomylný*** → ***důležitoneomylný***
important.ADJ + -o- + infallible.ADJ → *important-infallible.ADJ

Data set and evaluation

What we had to work with

Data set: Manually annotated data

- 1,447 manually annotated compounds (by us)
- extracted from DeriNet
- 80/10/10 split (1,158/145/144; training/testing/validation)

ryb-o-lov $\rightarrow$ ***ryba*** ***lov***
fishery.NOU fish.NOU hunt.NOU

Synthetic data

- 280,000 compounds synthesized using lexemes from DeriNet
- Used only for Czech Compound Splitter training

Adjective 1 + ***-o-*** + **Adjective 2** $\rightarrow$ **Compound Adjective**
důležitý + ***-o-*** + ***neomylný*** $\rightarrow$ ***důležitoneomylný***
important.ADJ + -o- + infallible.ADJ $\rightarrow$ *important-infallible.ADJ

Evaluation

How we evaluated performance

Metrics:

- **Accuracy**: Ratio of correct predictions vs. incorrect predictions
  - Applicable to both compound splitting and identification
  - Within splitting, a prediction is correct iff *all* parents are correct (unless stated otherwise)
- **Root accuracy**: Ratio of predicted parents within the correct derivational family
  - Only applicable to compound splitting
  - Checked using DeriNet

Evaluation

How we evaluated performance

Metrics:

- **Accuracy**: Ratio of correct predictions vs. incorrect predictions
  - Applicable to both compound splitting and identification
  - Within splitting, a prediction is correct iff *all* parents are correct (unless stated otherwise)
- **Root accuracy**: Ratio of predicted parents within the correct derivational family
  - Only applicable to compound splitting
  - Checked using DeriNet

Evaluation

How we evaluated performance

Metrics:

- **Accuracy**: Ratio of correct predictions vs. incorrect predictions
  - Applicable to both compound splitting and identification
  - Within splitting, a prediction is correct iff *all* parents are correct (unless stated otherwise)
- **Root accuracy**: Ratio of predicted parents within the correct derivational family
  - Only applicable to compound splitting
  - Checked using DeriNet

Evaluation

How we evaluated performance

Metrics:

- **Accuracy**: Ratio of correct predictions vs. incorrect predictions
  - Applicable to both compound splitting and identification
  - Within splitting, a prediction is correct iff *all* parents are correct (unless stated otherwise)
- **Root accuracy**: Ratio of predicted parents within the correct derivational family
  - Only applicable to compound splitting
  - Checked using DeriNet

Evaluation

How we evaluated performance

Metrics:

- **Accuracy**: Ratio of correct predictions vs. incorrect predictions
  - Applicable to both compound splitting and identification
  - Within splitting, a prediction is correct iff *all* parents are correct (unless stated otherwise)
- **Root accuracy**: Ratio of predicted parents within the correct derivational family
  - Only applicable to compound splitting
  - Checked using DeriNet

Evaluation

How we evaluated performance

Metrics:

- **Accuracy**: Ratio of correct predictions vs. incorrect predictions
  - Applicable to both compound splitting and identification
  - Within splitting, a prediction is correct iff *all* parents are correct (unless stated otherwise)
- **Root accuracy**: Ratio of predicted parents within the correct derivational family
  - Only applicable to compound splitting
  - Checked using DeriNet

Evaluation

How we evaluated performance

Metrics:

- **Accuracy**: Ratio of correct predictions vs. incorrect predictions
  - Applicable to both compound splitting and identification
  - Within splitting, a prediction is correct iff *all* parents are correct (unless stated otherwise)
- **Root accuracy**: Ratio of predicted parents within the correct derivational family
  - Only applicable to compound splitting
  - Checked using DeriNet

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

Our solutions

Three approaches to the tasks at hand

- Baseline solution
  - Naive explicit solution contextualizing the other two
  - Limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- *IML*()-based heuristic algorithm
  - Explicit solution based on a phonological similarity function (*IML*())
  - Also limited to compound splitting
  - Returns two parents
  - Requires a lexicon
- Czech Compound Splitter
  - Deep learning solution
  - Performs both compound identification and splitting
  - Returns any number of parents
  - Does not require a lexicon

IML() matrix function

The phonological string similarity function

The *IML*() (*Interlexical Matrices of Likeness*) function returns the phonological similarity of two Czech graphical words using a manually defined *correspondence matrix*.

- $IML(word_1, word_2) = 0$ iff $word_1 = word_2$
- $IML(word_1, word_2)$ need not equal $IML(word_2, word_1)$

The IML()-based heuristic algorithm tests all word-pairs $((lexeme_1, lexeme_2))$ from the lexicon that satisfy a heuristic filter that discards all obviously wrong word-pairs.

- otherwise, $800,000^2 = 6.4 \times 10^{11}$ operations

IML() matrix function

The phonological string similarity function

The *IML*() (*Interlexical Matrices of Likeness*) function returns the phonological similarity of two Czech graphical words using a manually defined *correspondence matrix*.

- $IML(word_1, word_2) = 0$ iff $word_1 = word_2$
- $IML(word_1, word_2)$ need not equal $IML(word_2, word_1)$

The IML()-based heuristic algorithm tests all word-pairs $((lexeme_1, lexeme_2))$ from the lexicon that satisfy a heuristic filter that discards all obviously wrong word-pairs.

- otherwise, $800,000^2 = 6.4 \times 10^{11}$ operations

IML() matrix function

The phonological string similarity function

The *IML*() (*Interlexical Matrices of Likeness*) function returns the phonological similarity of two Czech graphical words using a manually defined *correspondence matrix*.

- $IML(word_1, word_2) = 0$ iff $word_1 = word_2$
- $IML(word_1, word_2)$ need not equal $IML(word_2, word_1)$

The IML()-based heuristic algorithm tests all word-pairs $((lexeme_1, lexeme_2))$ from the lexicon that satisfy a heuristic filter that discards all obviously wrong word-pairs.

- otherwise, $800,000^2 = 6.4 \times 10^{11}$ operations

IML() matrix function

The phonological string similarity function

The *IML*() (*Interlexical Matrices of Likeness*) function returns the phonological similarity of two Czech graphical words using a manually defined *correspondence matrix*.

- $IML(word_1, word_2) = 0$ iff $word_1 = word_2$
- $IML(word_1, word_2)$ need not equal $IML(word_2, word_1)$

The IML()-based heuristic algorithm tests all word-pairs $((lexeme_1, lexeme_2))$ from the lexicon that satisfy a heuristic filter that discards all obviously wrong word-pairs.

- otherwise, $800,000^2 = 6.4 \times 10^{11}$ operations

IML() matrix function

The phonological string similarity function

The *IML*() (*Interlexical Matrices of Likeness*) function returns the phonological similarity of two Czech graphical words using a manually defined *correspondence matrix*.

- $IML(word_1, word_2) = 0$ iff $word_1 = word_2$
- $IML(word_1, word_2)$ need not equal $IML(word_2, word_1)$

The IML()-based heuristic algorithm tests all word-pairs $((lexeme_1, lexeme_2))$ from the lexicon that satisfy a heuristic filter that discards all obviously wrong word-pairs.

- otherwise, $800,000^2 = 6.4 \times 10^{11}$ operations

IML() matrix function

The phonological string similarity function

The *IML*() (*Interlexical Matrices of Likeness*) function returns the phonological similarity of two Czech graphical words using a manually defined *correspondence matrix*.

- $IML(word_1, word_2) = 0$ iff $word_1 = word_2$
- $IML(word_1, word_2)$ need not equal $IML(word_2, word_1)$

The IML()-based heuristic algorithm tests all word-pairs $((lexeme_1, lexeme_2))$ from the lexicon that satisfy a heuristic filter that discards all obviously wrong word-pairs.

- otherwise, $800,000^2 = 6.4 \times 10^{11}$ operations

Czech Compound Splitter

The all-in-one Czech compound handling tool

Czech Compounds Splitter (*CCS*) performs compound splitting and compound identification in one step.

It is an implementation of the *Marian* translator framework by Microsoft.

It was trained on:

- 1,164 genuine compounds, with their splittings
- 280,000 synthetic compounds, with their splittings
- the near entirety of DeriNet's non-compounds, with their derivational parents

zakódovat → ***kódovat***
encode.VERB code.VERB

Czech Compound Splitter

The all-in-one Czech compound handling tool

Czech Compounds Splitter (*CCS*) performs compound splitting and compound identification in one step.

It is an implementation of the *Marian* translator framework by Microsoft.

It was trained on:

- 1,164 genuine compounds, with their splittings
- 280,000 synthetic compounds, with their splittings
- the near entirety of DeriNet's non-compounds, with their derivational parents

zakódovat → ***kódovat***
encode.VERB code.VERB

Czech Compound Splitter

The all-in-one Czech compound handling tool

Czech Compounds Splitter (*CCS*) performs compound splitting and compound identification in one step.

It is an implementation of the *Marian* translator framework by Microsoft.

It was trained on:

- 1,164 genuine compounds, with their splittings
- 280,000 synthetic compounds, with their splittings
- the near entirety of DeriNet's non-compounds, with their derivational parents

zakódovat	→	***kódovat***
encode.VERB		code.VERB

Czech Compound Splitter

The all-in-one Czech compound handling tool

Czech Compounds Splitter (*CCS*) performs compound splitting and compound identification in one step.

It is an implementation of the *Marian* translator framework by Microsoft.

It was trained on:

- 1,164 genuine compounds, with their splittings
- 280,000 synthetic compounds, with their splittings
- the near entirety of DeriNet's non-compounds, with their derivational parents

zakódovat → ***kódovat***
encode.VERB code.VERB

Czech Compound Splitter

The all-in-one Czech compound handling tool

Czech Compounds Splitter (*CCS*) performs compound splitting and compound identification in one step.

It is an implementation of the *Marian* translator framework by Microsoft.

It was trained on:

- 1,164 genuine compounds, with their splittings
- 280,000 synthetic compounds, with their splittings
- the near entirety of DeriNet's non-compounds, with their derivational parents

zakódovat → ***kódovat***
encode.VERB code.VERB

Czech Compound Splitter

The all-in-one Czech compound handling tool

Czech Compounds Splitter (*CCS*) performs compound splitting and compound identification in one step.

It is an implementation of the *Marian* translator framework by Microsoft.

It was trained on:

- 1,164 genuine compounds, with their splittings
- 280,000 synthetic compounds, with their splittings
- the near entirety of DeriNet's non-compounds, with their derivational parents

$$\underset{\text{encode.VERB}}{\textit{zakódovat}} \rightarrow \underset{\text{code.VERB}}{\textit{kódovat}}$$

Czech Compound Splitter

The all-in-one Czech compound handling tool

Czech Compounds Splitter (*CCS*) performs compound splitting and compound identification in one step.

It is an implementation of the *Marian* translator framework by Microsoft.

It was trained on:

- 1,164 genuine compounds, with their splittings
- 280,000 synthetic compounds, with their splittings
- the near entirety of DeriNet's non-compounds, with their derivational parents

zakódovat → *kódovat*
encode.VERB code.VERB

Czech Compound Splitter

The all-in-one Czech compound handling tool

Czech Compounds Splitter (*CCS*) performs compound splitting and compound identification in one step.

It is an implementation of the *Marian* translator framework by Microsoft.

It was trained on:

- 1,164 genuine compounds, with their splittings
- 280,000 synthetic compounds, with their splittings
- the near entirety of DeriNet's non-compounds, with their derivational parents

zakódovat $\rightarrow$ ***kódovat***
encode.VERB code.VERB

Czech Compound Splitter

The all-in-one Czech compound handling tool

Czech Compounds Splitter (*CCS*) performs compound splitting and compound identification in one step.

It is an implementation of the *Marian* translator framework by Microsoft.

It was trained on:

- 1,164 genuine compounds, with their splittings
- 280,000 synthetic compounds, with their splittings
- the near entirety of DeriNet's non-compounds, with their derivational parents

zakódovat $\rightarrow$ ***kódovat***
encode.VERB code.VERB

Czech Compounds Splitter's typical errors in compound splitting:

Compound	CCS *splitting*	**Correct splitting**
dlouhohořící	*dlouhohořící*	*dlouho hořící*
CCS returns the original string, performing no splitting.		
osmiramenný	*osm ramenný*	*osm rameno*
CCS returns a non-existing derivative of an existing word.		
bíločerný	*bílo černý*	*bílý černý*
CCS includes the interfix in one of the parents.		

Czech Compounds Splitter's typical errors in compound splitting:

Compound	CCS *splitting*	**Correct splitting**
dlouhohořící	*dlouhohořící*	*dlouho hořící*
CCS returns the original string, performing no splitting.		
osmiramenný	*osm ramenný*	*osm rameno*
CCS returns a non-existing derivative of an existing word.		
bíločerný	*bílo černý*	*bílý černý*
CCS includes the interfix in one of the parents.		

Czech Compounds Splitter's typical errors in compound splitting:

Compound	CCS *splitting*	**Correct splitting**
dlouhohořící	*dlouhohořící*	*dlouho hořící*
CCS returns the original string, performing no splitting.		
osmiramenný	*osm ramenný*	*osm rameno*
CCS returns a non-existing derivative of an existing word.		
bíločerný	*bílo černý*	*bílý černý*
CCS includes the interfix in one of the parents.		

Results

How the approaches ended up performing

Compound identification:

- Performed only by *Czech Compound Splitter*
- Evaluated on 144 compounds and 141 non-compounds (no information rate: 50.5%)
- Accuracy: 92%

Compound splitting:

Method	1st parent accuracy	2nd parent accuracy	Overall accuracy	Overall root accuracy
Baseline	22%	42%	11%	16%
IML()	42%	66%	24%	39%
CCS	61%	66%	54%	61%

Conclusion

- We have made the first attempt into the automatic splitting and identification of Czech compound words.
- We have developed three tools:
  - a naive baseline solution,
  - a heuristic algorithm based on *IML*(), a custom phonological similarity function
  - *Czech Compound Splitter*, a deep learning solution
- *Czech Compound Splitter* turns out to be the best solution, in terms of both performance and practicality.
- It has achieved an accuracy of 54% in compound splitting and an accuracy of 92% in compound splitting.
- Furthermore, it can also find the single *derivational* parent of a given Czech word, if it's not a compound.

Conclusion

- We have made the first attempt into the automatic splitting and identification of Czech compound words.
- We have developed three tools:
  - a naive baseline solution,
  - a heuristic algorithm based on *IML*(), a custom phonological similarity function
  - *Czech Compound Splitter*, a deep learning solution
- *Czech Compound Splitter* turns out to be the best solution, in terms of both performance and practicality.
- It has achieved an accuracy of 54% in compound splitting and an accuracy of 92% in compound splitting.
- Furthermore, it can also find the single *derivational* parent of a given Czech word, if it's not a compound.

Conclusion

- We have made the first attempt into the automatic splitting and identification of Czech compound words.
- We have developed three tools:
  - a naive baseline solution,
  - a heuristic algorithm based on *IML*(), a custom phonological similarity function
  - *Czech Compound Splitter*, a deep learning solution
- *Czech Compound Splitter* turns out to be the best solution, in terms of both performance and practicality.
- It has achieved an accuracy of 54% in compound splitting and an accuracy of 92% in compound splitting.
- Furthermore, it can also find the single *derivational* parent of a given Czech word, if it's not a compound.

Conclusion

- We have made the first attempt into the automatic splitting and identification of Czech compound words.
- We have developed three tools:
  - a naive baseline solution,
  - a heuristic algorithm based on *IML*(), a custom phonological similarity function
  - *Czech Compound Splitter*, a deep learning solution
- *Czech Compound Splitter* turns out to be the best solution, in terms of both performance and practicality.
- It has achieved an accuracy of 54% in compound splitting and an accuracy of 92% in compound splitting.
- Furthermore, it can also find the single *derivational* parent of a given Czech word, if it's not a compound.

Conclusion

- We have made the first attempt into the automatic splitting and identification of Czech compound words.
- We have developed three tools:
  - a naive baseline solution,
  - a heuristic algorithm based on *IML*(), a custom phonological similarity function
  - *Czech Compound Splitter*, a deep learning solution
- *Czech Compound Splitter* turns out to be the best solution, in terms of both performance and practicality.
- It has achieved an accuracy of 54% in compound splitting and an accuracy of 92% in compound splitting.
- Furthermore, it can also find the single *derivational* parent of a given Czech word, if it's not a compound.

Conclusion

- We have made the first attempt into the automatic splitting and identification of Czech compound words.
- We have developed three tools:
  - a naive baseline solution,
  - a heuristic algorithm based on *IML*(), a custom phonological similarity function
  - *Czech Compound Splitter*, a deep learning solution
- *Czech Compound Splitter* turns out to be the best solution, in terms of both performance and practicality.
- It has achieved an accuracy of 54% in compound splitting and an accuracy of 92% in compound splitting.
- Furthermore, it can also find the single *derivational* parent of a given Czech word, if it's not a compound.

Conclusion

- We have made the first attempt into the automatic splitting and identification of Czech compound words.
- We have developed three tools:
  - a naive baseline solution,
  - a heuristic algorithm based on *IML*(), a custom phonological similarity function
  - *Czech Compound Splitter*, a deep learning solution
- *Czech Compound Splitter* turns out to be the best solution, in terms of both performance and practicality.
- It has achieved an accuracy of 54% in compound splitting and an accuracy of 92% in compound splitting.
- Furthermore, it can also find the single *derivational* parent of a given Czech word, if it's not a compound.

Conclusion

- We have made the first attempt into the automatic splitting and identification of Czech compound words.
- We have developed three tools:
  - a naive baseline solution,
  - a heuristic algorithm based on *IML*(), a custom phonological similarity function
  - *Czech Compound Splitter*, a deep learning solution
- *Czech Compound Splitter* turns out to be the best solution, in terms of both performance and practicality.
- It has achieved an accuracy of 54% in compound splitting and an accuracy of 92% in compound splitting.
- Furthermore, it can also find the single *derivational* parent of a given Czech word, if it's not a compound.

Acknowledgments

This work was supported by the Grant No. GA19-14534S of the Czech Science Foundation, the LINDAT/CLARIAH-CZ project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2018101), and by the Grant No. START/HUM/010 of Grant schemes at Charles University (Reg. No. CZ.02.2.69/0.0/0.0/19_073/0016935).

References


Introducing Linguistic Morphology.
Bauer L, 2nd edn. Washington, DC: Georgetown University Press, 2003.


Tvoření slov skládáním.
Bozděchová I. Nakladatelství Institut sociálních vztahů, 1994.


Je možná nová klasifikace českých kompozit?.
Štichauer P. *Časopis pro moderní filologii* 95(2), 2003.


The classification of compounds.
Bisetto K, Scalise, S. *Lingue e linguaggio* 4(2), 2005.


Kompozitnost' a kompozícia: príspevok k charakteristike zložených slov na materiáli západoslovanských jazykov.
Ološtiak M, Vojteková M. *Slovo a Slovesnost* 82(2), 2021.


Derivancze — derivational analyzer of Czech.
Pala, K, Šmerk, P. In: *International Conference on Text, Speech, and Dialogue*, 2015.


DeriNet 2.1.

Vidra, J,Žabokrtský, Z, Kyjánek, L, Ševčíková, M,Dohnalová, Š, Svoboda, E, Bodnár, J. URL:

http://hdl.handle.net/11234/1-3765.

References 2


GermaNet – a lexical-semantic net for German.
Hamp, B and Feldweg, H. In: *Automatic information extraction and building of lexical semantic resources for NLP applications*, 1997.


The CELEX lexical database.
Baayen, H, Piepenbrock, R, and Gulikers, L. Linguistic Data Consortium, 1996.


A Joint Approach to Compound Splitting and Idiomatic Compound Detection.
Krotova, I, Aksenov, S, Artemova E. In: *Proceedings of the 12th Language Resources and Evaluation Conference*, 2020.


Sanskrit word segmentation using character-level recurrent and convolutional neural networks.
Hellwig, O, Nehrdich, S. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.


Splitting of compound terms in non-prototypical compounding languages.
Clouet, E. and Daille, B. In: *Workshop on Computational Approaches to Compound Analysis*, 2014

Thank you for your attention!

Emil Svoboda

svoboda@ufal.mff.cuni.cz