From Electrophoresis to Wikidata: Festschrift in honor of Pierre Nugues

From Electrophoresis to Wikidata: Festschrift in honor of Pierre Nugues

Edited by Richard Johansson

April, 2025

Front-cover photo: Charlotte Christensen-Nugues

First page photo: From SVT Sydnytt, October 6, 2005

Published by University of Tartu Library, Estonia

ISBN: 978-9908-53-286-8

Volume Editor: Richard Johansson

Preface

This Festschrift is a tribute to professor Pierre Nugues on the occasion of his 65th birthday.

As a professor of Computer Science at the Faculty of Engineering (LTH) at Lund University, Pierre Nugues has made wide-ranging scientific contributions throughout his career. His PhD dissertation was on image processing techniques for the interpretation of two-dimensional electrophoresis gels (*Interprétation de gels d'électrophorèses bidimensionnelles*, University of Nancy, 1989), but Pierre is known primarily as a contributor to the field of natural language processing (NLP), where he has been active since the early 1990s. His work in this area has always had a practical angle and has often focused on questions related to semantic processing of language. He has made research contributions on fundamental NLP problems such as knowledge extraction, named entity recognition and linking, coreference resolution, and semantic role labeling, and applications such as question answering, dialogue systems, virtual reality therapy and computer-assisted language learning. The research carried out by his team has resulted in the implementation of multilingual NLP tools for tasks such as semantic parsing and named entity linking, which have been widely used and are available as open-source software packages. He has a passion for community-driven projects such as Wikidata and Wikipedia (and is a frequent contributor to Wikipedia), and much of his recent research has been about how NLP technologies can make use of these open resources, and how older lexicons and encyclopedias (such as *Nordisk Familjebok*, Diderot's *Encyclopédie*, and the *Petit Larousse illustré*) can be linked to modern resources. His wide-ranging contributions are reflected in the title of the volume, *From Electrophoresis to Wikidata*.

Beyond his research, Pierre is also committed to teaching, and he has taught courses in NLP, machine learning and other computer science courses at LTH for many years. He is the author of two textbooks for students of natural language processing: *An Introduction to Language Processing with Perl and Prolog* and more recently *Python for Natural Language Processing: Programming with NumPy, scikit-learn, Keras, and PyTorch*, both of which provide hands-on implementations of many well-known models and algorithms used in NLP. These books have been used at his NLP courses at LTH over the years.

We are honored to present this Festschrift in recognition of Pierre's contributions over his career. The *Tabula Gratulatoria* in this volume includes the names of people in Lund and elsewhere who wish to congratulate him on his birthday. The seven papers included in the Festschrift reflect the breadth of his interests, and they have been contributed by people who have interacted with him over the years. While there are unfortunately no contributions on the interpretation of electrophoresis gels, the papers are on a variety of scientific and pedagogical topics connected to his work. We hope that Pierre will enjoy reading these papers!

The editor
April, 2025

Tabula Gratulatoria

Lars Ahrenberg

Per Andersson

Lars Bendix

Anders Björkelund

Markus Borg

Lars Borin

Peter Exner

Niklas Fors

Jonas Granfeldt

Jakob Grundström

Mathias Haage

Love Hafdell

Görel Hedin

Thore Husfeldt

Richard Johansson

Arne Jönsson

Håkan Jonsson

Martin Georg Ljungqvist

Jacek Malec

Boris Magnusson

Dennis Medved

Joakim Nivre

Aarne Ranta

Björn Regnell

Per Runeson

Emma Söderberg

Elin A. Topp

Rebecka Weegar

Table of Contents

A Tribute to my Former Teacher

Anders Björkelund
Centre for Environmental and Climate Science
Computational Science for Health and Environment
Lund University, Sweden
anders.bjorkelund@cec.lu.se

1 Introduction

My first interaction with Pierre was as a student in his class on NLP given at the CS department in Lund. Prior to that I had never even heard of language processing, let alone pondered over the potential challenges. The class was exciting and different, with a high pace basically touching everything NLP-related in the lectures, a variety of labs, as well as a final programming project with both written report and an oral presentation. I recall how this stood out from many other classes I had taken in Lund and it was a lot of fun. I now realize that this ambitious setup probably implied more work than the bare minimum for Pierre, but as a student i profited considerably.

2 Working with Pierre

After completing Pierre's NLP class I ended up doing my master's thesis with him during spring 2009. We did the CoNLL 2009 Shared Task. It was a tight timeline with a high pace. Pierre kept asking what were the latest numbers, how is the development going. The code was a nasty combination involving C, Java, Perl, and a mess of shell scripts. In parallel, Pierre also mentioned from time to time that conference, to be held in summer, and that I should go there to present our work. I had no intention of going anywhere. Eventually we submitted our runs, the shared task closed, and we did quite alright and wrote up the paper. Later that summer I also attended that conference and presented the work. This was a rather significant moment in my life, gaining a new perspective both on the breadth of the NLP/CL community as well as what an academic career implies.

After an exciting summer of 2009 Pierre asked me to do work in a robotics project, supposedly involving NLP. I was ambivalent but unemployed after completing my degree. The reasons for my reluctance was two-fold. I had an urge to get out of Lund, I wanted a change of environment. Second, having encountered the NLP community at that conference, I had learned that at other universities there are full groups or even departments working on language processing. Doing a PhD in such an environment was attractive. In hindsight, this makes Pierre's endeavor in Lund even more impressive – maintaining the NLP education at undergraduate and graduate level as a single faculty.

I stayed with Pierre for some time, working as developer in said robotics project, considering to apply as a PhD student, eventually opting to pursue doctoral studies elsewhere. Before leaving we managed to learn a lot about industrial robotics and published a couple of papers on the topic. During that time, Pierre also pushed me into cleaning up the CoNLL code and making it open source. Said and done, published as a separate paper. To this day, these papers remain some of my most cited.

3 Conclusion

As I'm writing this up, my 8-year old daughter is asking me what I'm up to. I try to explain what a Festschrift is, that I'm writing a document regarding my former teacher. Trying to explain how one person can affect the winding paths of one's life to an 8-year old is tricky. To make things a bit more tangible to her, I tell her that "If it wasn't for Pierre, mommy and I would never have met." This is a true statement involving conferences and career paths, the underlying details being outside the scope of this document. In any case, I guess it is also a way of summarizing my gratitude towards my former advisor: Dear Pierre, I am deeply grateful that our paths crossed. That you believed in me and pushed me into doing some of those things I wasn't all that enthusiastic about at the time. I never did a PhD under your supervision, but my PhD and academic career would not have been possible without you. Thank you.

From Electrophoresis to Wikidata: Festschrift in honor of Pierre Nugues, pages 1–1.
Published by University of Tartu Library, May 2025.

Deltagande tar vi farväl av participen

Lars Borin
Språkbanken Text
Institutionen för svenska, flerspråkighet och språkteknologi
Göteborgs universitet
lars.borin@svenska.gu.se

Abstract

Saldo är Språkbankens centrala lexikala resurs för automatisk analys på ordnivå av modern svensk text. Version 3 av Saldo ska släppas under 2025, där de stora skillnaderna mot föregående version framför allt återfinns i den morfologiska beskrivningen, i synnerhet ordklassindelningen. I artikeln diskuteras och motiveras hur participen behandlas i Saldo 3.

1 Saldo 3 runt hörnet

Nästa version av Saldo beräknas komma ut i år. Saldo är den centrala lexikonresursen för modern svenska i Språkbankens lexikala forskningsinfrastruktur och har varit i bruk sedan 2008 som den lexikala motorn i lemmatiserings- och sammansättningsanalysfunktionen i Språkbankens korpusimportkedja (Borin et al., 2013, 2021).

Saldo 1 släpptes 2008, produktionsversionen av Saldo 2 (v. 2.3) kom 2015, och nu är det som sagt dags för nästa produktionsversion, Saldo 3.3, som beräknas komma ut i år, 2025.

Den stora nyheten och skillnaden mellan Saldo 2 och Saldo 3 återfinns i den morfologiska beskrivningen, närmare bestämt i ordklassindelningen. Allra märkbarast är skillnaden förmodligen när det gäller participen. Saldo 1 och 2 följer en traditionell svensk grammatisk beskrivningsmodell enligt vilken participen betraktas som böjningsformer av verb. Den praktiska konsekvensen vid lemmatisering blir att *lekande* i frasen *de lekande barnen* förs till verblexikoningången *leka* och att *kvävt* i *ett kvävt fniss* på samma sätt förs till verbet *kväva*. Participen kan alltså inte vara egna lexikonenheter i Saldo 2.

Ovanstående blir annorlunda i Saldo 3, där participen får egna lexikoningångar, både betydelser och lemgram.

Men vilken ordklass ska participen ha i Saldo 3 (eller vilka ordklasser, ifall presensparticipen och preteritumparticipen ska behandlas olika)?

2 Particip: en kortkort historia

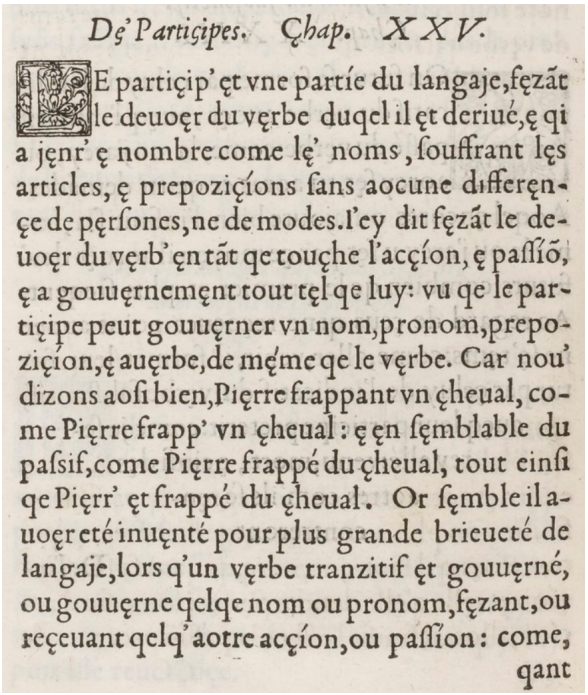
Genom hela vår grammatikhistoria har participen intagit en särställning. Den äldsta grekiska ordklassindelningen förde in participen under verben, men från och med Dionysios Thrax (som levde c:a 170–90 f.v.t.) sågs participen som en egen ordklass (Robins, 1979, 33). Detta tillstånd varade sen i ungefär 1500 år. På det stora hela anslöt sig den romerska grammatiktraditionen sömlöst till Thrax. I Priscianus latinska grammatik (c:a 500 v.t.) räknas participen också som en egen ordklass. Motiveringen är att participen delar egenskaper med både verb och substantiv (tempus- resp. kasusböjning)[1] och därför ska särskiljas från båda dessa (Robins, 1979, 57).

Under övergången mellan medeltid och renässans i samband med att grammatikor över folkspråken började sammanställas omorganiserades det antika ordklassystemet så att substantiv och adjektiv skildes åt och participen införlivades med verbböjningen (Robins, 1979, 110). Se t.ex. figur 1.

Den svenska grammatiktraditionen har som sagt övervägande behandlat participen som verbformer. Detta gäller såväl äldre (t.ex. Enberg, 1836; Almqvist, 1840) som nyare (t.ex. Thorell, 1973) arbeten. Nämnvärda undantag är framför allt de två grammatikorna som getts ut av Svenska Akademien, *Svenska Akademiens grammatik* (SAG; Teleman et al., 1999) och *Svenska Akademiens språklära* (SAS; Hultman, 2003). Mer om dem nedan.

[1] Därav benämningen *particip*: ”av lat. *participium*, till *particeps*, som deltager [. . .] Benämningen syftar på att participet har del i såväl verbets som adjektivets karaktär.” (SAOB, 1898–2023, s.v. *particip*)

From Electrophoresis to Wikidata: Festschrift in honor of Pierre Nugues, pages 2–6.
Published by University of Tartu Library, May 2025.



De' Partiçipes. Çhap. XXV.

IE partiçip' ęt vne partíe du langaje, fęzãt le deuoęr du vęrbe duqel il ęt deriué, ę qi a jęnr' ę nombre come lę' noms, ſouffrant lęs articles, ę prepoziçíons ſans aocune differęnçe de pęrſones, ne de modes. I'ey dit fęzãt le deuoęr du vęrb' ęn tãt qe touçhe l'acçíon, ę paſſíõ, ę a gouuęrnemęnt tout tęl qe luy: vu qe le partiçipe peut gouuęrner vn nom, pronom, prepoziçíon, ę auęrbe, de męme qe le vęrbe. Car nou' dizons aoſi bien, Pięrre frappant vn çheual, come Pięrre frapp' vn çheual: ę ęn ſęmblable du paſsif, come Pięrre frappé du çheual, tout einſi qe Pięrr' ęt frappé du çheual. Or ſęmble il auoęr eté inuęnté pour plus grande brieueté de langaje, lors q'un vęrbe tranzitif ęt gouuęrné, ou gouuęrne qelqe nom ou pronom, fęzant, ou reçeuant qelq' aotre acçíon, ou paſſíon: come,

qant

Figur 1: Participen i en tidig fransk grammatik (Meigręt, 1550, 100v)

På webbplatsen Svenska.se[2] kan man slå i de tre ordböcker som Svenska Akademien finansierar, *Svenska Akademiens ordlista* (SAOL 14, 2015) och *Svensk ordbok utgiven av Svenska Akademien* (SO, 2021) samt den delvis historiska *Svenska Akademiens ordbok* (SAOB, 1898–2023). SAOL:s onlineversion på Svenska.se visar samtliga böjningsformer för alla böjliga uppslagsord. Där ingår participen i verbens böjningsparadigm.

Saldo används som sagt för lexikalisk analys vid Språkbankens korpusimport. Den samverkar därvid med en ordklasstaggare tränad på SUC-korpusen (Ejerhed et al., 1992), en svensk referenskorpus med en miljon ord från 1990-talet med blandade genrer och manuellt kontrollerade lingvistiska annotationer (grundform, ordklass, morfosyntaktisk beskrivning och namnkategorisering). Till skillnad från Saldo separerar SUC-taggningen ut participen som egen huvudkategori komplett med egen grundform (t.ex. *lekande*, *kvävd*), något som gör interaktionen med Saldo-analysen något intrikatare än man skulle ha önskat.

Till sist kan vi notera ett faktum av växande betydelse för språkteknologi: i anslutning till initiativet *Universal Dependencies*[3] (UD) har en uppsättning universella ordklasstaggar (”universal part-of-speech tags”) definierats, f.n. 17 stycken (de Marneffe et al., 2021, 261). Där ingår inte participen som egen ordklass, och inga närmare anvisningar ges om hur participen ska annoteras[4].

3 Från ordklass till verbform till ordklass igen

I svensk grammatiktradition har participen således levt ett slags superpositionsliv, som en ”Schrödingers kategori” frestas man säga. De behandlas typiskt som böjningsformer av verbet, men grammatikförfattarna verkar alltid känna att participen ändå måste omnämnas i samband med att adjektiven beskrivs. Detta är särskilt påtagligt i SAS, där participen å ena sidan behandlas under adjektiven, men där preteritumparticipen (men inte presensparticipen) listas i verbens böjningsmönster.

Med SAG kan man säga att en mer än tvåtusenårig cirkel har slutits. Där har nämligen participen (återigen) fått status av egen ordklass. Det motiveras så här:

> Utöver de tre stora klasserna av böjliga ord [substantiv, adjektiv och verb] finns det ett antal andra ordklasser som karakteriseras av böjning och även i övrigt uppvisar likheter med de stora klasserna men som samtidigt kännetecknas av säregenskaper som gör det praktiskt att redovisa dem som egna klasser. [...] Presensparticip och perfektparticip är avledningar av verb. I böjningshänseende fungerar participet som ett adjektiv. [...] Också syntaktiskt fungerar i stort sett participen som adjektiv, dvs. som som attribut, predikativ och adbverbial. Participen kan emellertid ändå behålla vissa av verbets syntaktiska egenskaper och konstrueras t.ex. med objekt friare än adjektiven[.] (Teleman et al., 1999, 9–11)
>
> [...]
>
> Participen är regelbundet avledda av verb och har därför traditionellt ofta redovisats som böjningsformer av verben. De konstrueras också ofta med samma bestämningar som de verb de är avledda från. Syntaktiskt fungerar de dock i huvudsak som adjektiv och perfektparticipet kongruensböjs också som ett adjektiv. På grund av denna likhet med två andra ordklasser redovisas participen i denna grammatik i enlighet med en äldre grammatisk tradition som en självständig ordklass. (Teleman et al., 1999, 583)

4 Participet: ordklassernas Pluto?

Internationella astronomiska unionen antog år 2006 en definition av termen *planet*, som fick till följd att Pluto ändrade klassificering från planet till

[2] https://svenska.se

[3] https://universaldependencies.org/

[4] https://universaldependencies.org/u/feat/VerbForm.html

dvärgplanet[5], för att den helt enkelt inte uppfyllde de nu explicitgjorda kriterierna på vad som ska få kallas ”planet”.

Liksom Pluto enligt en rad kriterier uppfattades som en avvikare bland de tidigare nio planeterna i vårt solsystem, får man en liknande känsla av SAG:s participordklass, och i samband med omarbetandet av Saldomorfologin verkade det lämpligt att försöka styrka eller förkasta den intuitionen.

Otvetydigt är det därför att det är en avvikare bland ordklasserna som participet behandlas speciellt i grammatikor. Frågan är bara om detta motiverar att participen skulle tilldelas ordklasstatus. Svaret blir i Saldos fall ”nej”: participet är inte en planet. Men: det är inte en dvärgplanet heller.

Den nästan hundraprocentiga produktiviteten är antagligen en faktor som gör att particip uppfattas som verbböjning, medan ordklasskiftet är typiskt för avledning. Man noterar dock att liknande resonemang inte förs när det gäller verbalsubstantiv, i synnerhet dem på *-(a)nde*, som både är näst intill totalt produktiva och orsakar ordklassbyte, men som även i sin substantivhamn kan ta (en del av) utgångsverbets argument:

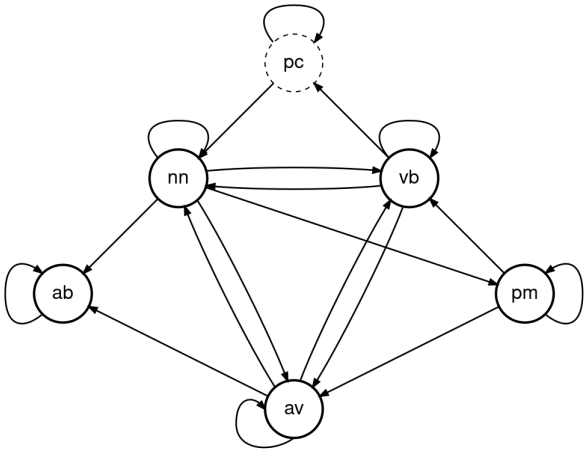
> Aktanter vid verb- och adjektivavledda substantiv är oftast desamma som vid respektive verb och adjektiv. *Till dessa substantiv ansluter sig sådana substantiv som har liknande betydelse men som inte är morfologiskt besläktade med verb och adjektiv.* (Teleman et al., 1999, 31–32; kursivering tillagd)

Om dessa verbalsubstantiv kan föras in under substantiven, borde nog de verbaladjektiv som participen ju faktiskt är kunna klassas som adjektiv, eftersom det även i det fallet redan finns medlemmar av den kategorin med verbliknande valens:

> Många adjektiv är flerställiga, dvs. har en betydelse som förutsätter mer än en aktant. Andra aktanter än predikatiosnbasen kan då anges med komplement som vanligen är adverbial (t.ex. *rädd för hissar*, *noggrann med föreskrifterna*) men som vid ett fåtal adjektiv kan vara objekt (t.ex. *trogen idealet*). Några adjektiv har (med vissa undantag) syntaktiskt obligatoriskt komplement, t.ex. *lik* med obligatoriskt objekt (*lik sin mor*), *benägen* med obligatoriskt adverbial (*benägen för lättja*). Vid de flesta adjektiv är komplementen syntaktiskt optionella. (Teleman et al., 1999, 166)

När det gäller participens eventuella status som egen ordklass och i synnerhet i det aktuella fallet, som ju handlar om (Saldos) morfologi, alltså

[5] https://www.iau.org/lau/Publications/List-of-Resolutions



Figur 2: Avledningsrelationer mellan öppna ordklasser i svenska. Heldragen kontur: ordklassen innehåller oavledda ord (ordklassetiketter: ab: adverb; av: adjektiv; nn: substantiv; pc: particip; pm: egennamn; vb: verb)

formegenskaper och inte i första hand betydelse inklusive funktion, förefaller det som framgår ur figur 2 och tabell 1 högst relevant.

I figur 2 och tabell 1 visas hur SAG:s sex öppna ordklasser substantiv (nn), egennamn (pm), adjektiv (av), adverb (ab), verb (vb) och particip (pc) är relaterade ordbildningsmässigt: pilarna anger vilka kombinationer av utgångsordklass och deriverad ordklass som är möjliga. Där syns tre tätt sammanlänkade centrala ordklasser (nn, av, vb) där alla möjligheter utnyttjas i språket.

De övriga tre klasserna beter sig avvikande på olika sätt i jämförelse:

- adverben kan vara men ger inte upphov till avledningar (utom sporadiskt inom ordklassen själv)
- egennamnen ger regelmässigt upphov till men är bara sporadiskt avledningar
- participen är (alltid) men ger sällan upphov till avledningar

I figur 2 illustreras även en avgörande faktor för vårt fall: bland adverben och egennamnen (och även i de tre kärnordklasserna) finns en mängd oavledda ord (t.ex. *väl*, *fort*, *Pierre*, *Caen*), medan participen enbart omfattar avledningar (*förstörd*, *ostörd*), vilket i figur 2 markeras medelst participnodens brutna konturlinje. Om ett ord skulle råka ha formen av ett particip, men sakna grundverb, innebär det automatisk klassning som adjektiv.

Ur ett typologiskt perspektiv kan även anföras att particip historiskt sett verkar uppkomma genom analogisk utvidgning av semantiken hos

⇒ **nn**	(Teleman et al., 1999, 34)
av	*jämnhet*, *nykterist*
nn	*vänskap*, *otur*
pc	*bundenhet*, *havandeskap*
pm	*platonism*, *närking*
vb	*skrivande*, *konsument*
⇒ **vb**	(Teleman et al., 1999, 516–517)
av	*smalna*, *modernisera*
nn	*bila*, *befolka*
pm	*wallraffa*
vb	*begå*, *förse*
⇒ **av**	(Teleman et al., 1999, 177–178)
av	*sjuklig*, *otålig*
nn	*manlig*, *geografisk*
pm	*Tegnérsk*, *göteborgsk*
vb	*skadlig*, *vridbar*
⇒ **pc**	(Teleman et al., 1999, 594)
pc	*förstörd*, *respekterad*
vb	*oförstörd*, *ostörd*
⇒ **ab**	(Teleman et al., 1999, 631)
ab	*ogärna*, *olika*
av	*säkerligen*, *grundligen*
nn	*nationsvis*, *gradvis*
⇒ **pm**	(Teleman et al., 1999, 124)
nn	*Lundén*, *Steninge*
pm	*Gunsan*

Tabell 1: Avledningsrelationer mellan öppna ordklasser i svenska med exempel (ordklassetiketter: ab: adverb; av: adjektiv; nn: substantiv; pc: particip; pm: egennamn; vb: verb)

denominala adjektiv bildade av verbalsubstantiv (Haspelmath, 1994, 170), samt att det inte finns något kristallklart kriterium (eller mängd av kriterier) för att dra en klar gränslinje mellan particip och adjektiv, där de ligger längs ett ordklasskontinuum som sträcker sig mellan verb och substantiv (Haspelmath, 1994, 171–172).

Slutligen väger det morfologiska argumentet tungt i det här sammanhanget, eftersom det gäller just morfologin i Saldo, en lexikalisk resurs som ska fungera i ett större språkteknologiskt sammanhang: participen böjs exakt som adjektiv och participen ingår som sagt inte i UD:s ordklassuppsättning.

5 Slutsats

Arbetet med Saldo 3 har lett till önskan att behandla participen som egna lexikonenheter, men även till insikten att SAG:s participordklass på sin höjd är en slags ”dvärgordklass” (se diskussionen ovan). Eftersom det ju redan finns en ordklass där participen kan ingå utan att den ordklassen behöver definieras om, varför inte helt enkelt klassificera participen som adjektiv? Därmed blev beslutet att ta avsked av SAG:s ordklass particip men inte att behålla den egendomliga lösningen att låta en viss – om än högproduktiv – avledning få rangen av böjningsform som i traditionell svensk grammatik. Istället välkomnas participen i adjektivens krets i Saldo. Där bör de känna sig hemma, eftersom det redan finns oavledda adjektiv med de egenskaper som brukar anses vara betecknande för participen.

Efterord

Jag är mycket glad över att ha fått möjligheten att bidra till en festskrift för Pierre Nugues, som jag har känt i många år och haft att göra med i ännu fler år: första gången vi möttes var för kanske 30 år sedan i Uppsala, där jag arbetade då och där Pierre sökte en anställning. Anställningen gick till en annan sökande, men det får nog sägas ha varit Uppsalas förlust och Lunds vinst. Vi har tyvärr bara samarbetat närmare en gång, i vårt gemensamma kulturomikprojekt, ett framgångsrikt samarbete som jag minns med glädje.

Grattis på födelsedagen, Pierre!

References

Carl Jonas Love Almqvist. 1840. *Svensk språklära*, 3 uppl. M. Wirsells förlag, Stockholm.

Lars Borin, Markus Forsberg, Lennart Lönngren. 2013. SALDO: A touch of yin to WordNet’s yang. *Language Resources and Evaluation*, 47(4):1191–1211.

Lars Borin, Markus Forsberg, Lennart Lönngren, Niklas Zechner. 2021. Swedish FrameNet++: Lexical samsara. I: Dana Dannélls, Lars Borin, Karin Friberg Heppin, red., *The Swedish FrameNet++: Harmonization, Integration, Method Development and Practical Language Technology Applications*, pages 69–95. John Benjamins, Amsterdam.

Eva Ejerhed, Gunnel Källgren, Ola Wennstedt, Magnus Åström. 1992. The linguistic annotation system of the Stockholm–Umeå corpus project: Description and guidelines. Teknisk rapport. Umeå: Institutionen för lingvistik, Umeå universitet.

Lars Magnus Enberg. 1836. *Svensk språklära utgifven av Svenska Akademien*. A.G. Hellsten, Stockholm.

Martin Haspelmath. 1994. Passive participles across languages. I: Barbara Fox Paul J. Hopper, red., *Voice: Form and function*, pages 151–177. John Benjamins, Amsterdam.

Tor G. Hultman. 2003. *Svenska Akademiens språklära*. Norstedts, Stockholm.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, Daniel Zeman. 2021. Universal dependencies. *Computational Linguistics*, 47(2):255–308.

Louís Meigret. 1550. *Le tretté de la grammere françoeze, fet par Louís Meigret Líonoes*. Chrestien Wechel, Paris. https://gallica.bnf.fr/ark:/12148/btv1b8624665r.image.

Robert Henry Robins. 1979. *A short history of linguistics*, 2 uppl. Longman, London.

SAOB. 1898–2023. *Svenska Akademiens ordbok*. Gleerups, Lund.

SAOL 14. 2015. *Svenska Akademiens ordlista*, 14 uppl. Norstedts, Stockholm.

SO. 2021. *Svensk ordbok utgiven av Svenska Akademien*, 2 uppl. Svenska Akademien, Stockholm.

Ulf Teleman, Staffan Hellberg, Erik Andersson. 1999. *Svenska Akademiens grammatik: 2 Ord*. Norstedts, Stockholm.

Olof Thorell. 1973. *Svensk grammatik*, 2 uppl. Esselte Studium, Stockholm.

The Nugues Approach to Project Courses

Görel Hedin
Department of Computer Science
Lund University
gorel.hedin@cs.lth.se

Abstract

Project courses at the Master of Science level is an excellent opportunity for students to do a small research-style project as a preparation for the Master thesis. Such projects can even lead to publications, as shown multiple times by Pierre Nugues in his supervision of student projects.

In this paper I describe the Nugues approach to student projects and how I use it for compiler projects. I also discuss IPSERC, an article structure I find useful for computer science students, to complement the IMRaD structure they usually learn in scientific writing classes.

1 Introduction

The field of computer science evolves rapidly, and students taking advanced courses are often eager to try out their new knowledge, constructing some new experimental tool, investigating a new technique, or solving some real-world problem. Doing this in the context of a research-oriented project course gives students an excellent preparation for their thesis project.

In this paper, I will describe the *Nugues approach*, i.e., Pierre Nugues's approach to running such projects. Pierre developed this approach for his projects in language technology, and I will also discuss how I use it for compiler projects, and how it came about that I started running such projects at all.

Key ideas in the Nugues approach include getting students started on both implementation and writing very quickly, and with weekly feedback on progression. This is important not the least as the department project course is short, running during a seven week study period. Another important part of the approach is that students write up a short research-style paper, and present it in a conference-like setting. Pierre even makes it an explicit goal for his students to potentially submit the paper to a real conference. I find this very inspiring, in particular since he has published several papers with his students based these projects, see, e.g., Grundström and Nugues (2014), Weegar et al. (2014), and Norrby and Nugues (2015).

A difficulty in supervising computer science students is that they often do not know what or how to write. Their training in scientific writing is typically focused on the IMRaD structure for experiments in medicine and natural science, with Introduction, Methods, Results, and Discussion. In the computer science field much of the research is focused on creating solutions to problems, and the IMRaD structure does not fit. This mismatch is something the Nugues approach helps counter: the students get to see typical computer science papers with sections on solutions and evaluation. I use the term *IPSERC* for this typical structure (Introduction, Problem, Solution, Evaluation, Related Work, Conclusion).

The reason that I and many others at our department came to run these student projects at all, was a bit peculiar. It was triggered by a bureaucratic decision that was taken by our faculty in 2013, namely that the department must give *fewer* courses than before. I describe this briefly in Section 2. Then I present the Nugues approach in Section 3, some adaptations I did for my compiler projects in Section 4, and the IPSERC structure and its relation to IMRaD in Section 5. Finally, I end with some concluding remarks in Section 6.

2 A bureaucratic decision

Working at the university is a constant struggle between trying to do good teaching and research, and trying to navigate the administration. In 2012, our faculty got a decrease in teaching funding, and at the same time there was an "overproduction" of

From Electrophoresis to Wikidata: Festschrift in honor of Pierre Nugues, pages 7–10.
Published by University of Tartu Library, May 2025.

student credits. I.e., the faculty was teaching more than it got paid for by the university (and in the end, by the government). The main reason for this overproduction was that the university had extended the engineering education from 4.5 years to 5 years, in order to adapt to the new Bologna model for education, and without being compensated for this by the government. The faculty decided, as one part in trying to solve the financial problems, to decrease the number of courses offered at the faculty, requiring all departments to reduce their number of ”course codes”, i.e., the number of courses offered. This did of course not change the number of credits that the students wanted to or needed to take, so I will leave any explanation of the logic behind this decision to future work.

This decision was very much at odds with our plans at the computer science department. The whole area was rapidly expanding worldwide, and we had too few advanced courses to offer an increasing number of interested students. Yet, even our department had to decrease the number of ”course codes”. How should we solve this?

At the time, we had three project courses: one in operating systems (EDAF01), one in intelligent systems (EDAN50), and one in language technology (EDAN60). The latter one was Pierre Nugues's course. We then got the brilliant idea to merge these courses into one new ”course code” (EDAN70). We could in practice continue teaching as before, but with a bit more administration as students would now sign up just for the project course, and we would need to spend some time on sorting them into topics.

Merging the project courses had an upside as well. Other instructors wanting to run student projects could easily join without having to set up their own course. I was one of them. I was teaching the compilers course, and wanted to start a course where students could do a compiler project. In the current situation, adding a new course was out of the question, but I could perhaps achieve something similar through the new merged project course.

3 The Nugues Approach

Since Pierre had been so successful in running his student projects, I interviewed him about his approach in the fall of 2014, before starting my first batch of projects. Here is what I noted down:

Prepared proposals. The instructor prepares project proposals beforehand, each with a short description, and presents them at an introductory meeting. Students can in general not propose their own project, but if someone has an idea, the instructor will consider turning it into a proper proposal.

Concrete product. Each project has a concrete goal: a product, on which some property can be measured.

Small groups. Students work either individually or in pairs.

Weekly deadlines. The instructor meets each student group every week and gives them concrete goals for the next week. Students demo results at each meeting.

Start reading and writing immediately. Already the first week, the students should write the first outline of their report, as well as read a related article.

Start implementing immediately. The students need to start the concrete implementation immediately, to get something running as soon as possible, no matter how small.

Baseline. Many projects are about improving efficiency or computing something with higher precision. The instructor helps the students find a suitable baseline for measurements.

Presentation before report. The course runs for 7 weeks, with presentations in week 7. The final report is then not due until a few weeks later, after the Christmas break.

Clear requirements. To pass the course, the students need to a) complete a 4-page double column research-style report (3 pages if they work individually), b) present the work in a 15 minute talk using 10 slides, and c) meet the weekly deadlines, producing what has been agreed on.

4 Adapting to compiler projects

The Nugues guidelines fitted very well with the compiler projects I had in mind, and I have run the compiler projects in this way since the start in 2014. One minor difference is that many of the compiler projects are about building new tools,

Project title
Short overall description
Illustration
3-4 Concrete steps/subgoals
Optional stretch goals
Evaluation suggestions
References to key articles
Name of supervisor

Table 1: Content of project proposal slide

Week 1	Introduction meeting, select project
Week 2	Report with draft intro and outline
Week 3	First demo
Week 4	First release of code
Week 5	Report extended with solution
Week 6	Draft of presentation slides
Week 7	Presentation at seminar
+4 weeks	Final report and code

Table 2: Weekly deadlines

sometimes interactive, and where there is no clear baseline to evaluate against. The important aspects to include in the evaluation can then be, for example, demonstration of that the approach works on interesting examples, and that the performance is sufficient for practical use.

In my research group, the PhD students had nice ideas for compiler projects, and to make things work in a streamlined way, I developed a template for a project proposal slide (Table 1) and a standardized set of weekly deadlines (Table 2). Each proposal includes a number of concrete steps to get going, and sometimes also stretch goals for the ambitious student. In addition to writing a project report, the compiler students also publish their code as open source.

5 IPSERC versus IMRaD

In medicin and natural science, the main goal of research is to find new knowledge about the existing world, typically following research methods to establish this knowledge. This fits well with the IMRaD structure of research articles. In engineering sciences, including computer science, the goal of research is often instead to invent new methods to perform a useful task. We might call such an invented method a *solution method* to distinguish it from a *research method*: A solution method performs some useful task whereas a research method helps establishing knowledge about some existing phenomenon. Of course, once a new solution method has been invented, it becomes part of the world. It is then interesting to analyze its properties, in which case the usual research methods can be useful.

Courses in scientific writing will usually point out the importance of looking at existing literature in the research field, to understand how to structure articles for that field. However, concrete advice is then typically focused on the IMRaD model only. This confuses engineering and computer science students, since IMRaD does not match the structure of the papers they read or what they are expected to write. For example, what is a ”method” in their research? Is it perhaps the way they developed their solution? Is it how they came up with the idea for their new solution? Or what is it?

To balance the omnipresent IMRaD structure, I therefore propose the IPSERC structure as a concrete article structure which many computer science and engineering papers follow. An IPSERC paper has an Introduction section that includes a Problem description. The meat of the paper is then a description of a suggested Solution to this problem, described in one or more sections. After this there is typically a section called Evaluation which describes various properties of the suggested solution. Often, the evaluation involves experiments, and IMRaD can then be useful for structuring the Evaluation section. After the Evaluation section there is often a Related Work section. While the Introduction will include a brief discussion of previous related work, the Related Work section can discuss similarities and differences in more detail, since it is placed after the Solution and Evaluation sections. Figure 1 summarizes the IPSERC structure and its relation to IMRaD.

I have found it very helpful to present this explicit IPSERC structure to computer science students. It makes it easier for them to recognize the structure of the papers they read, identifying both IPSERC and IMRaD elements in them, and making students more aware that different research areas have different traditions.

6 Concluding remarks

Seven weeks is a very short time to do both implementation, evaluation, and research-style report writing, but with the Nugues approach, it works

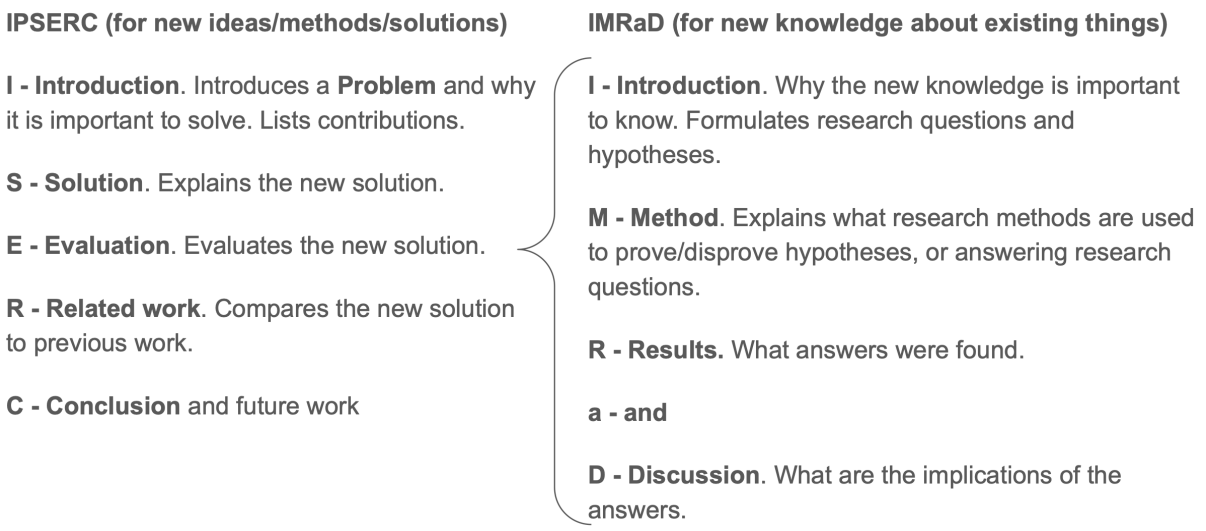


Figure 1: Typical IPSERC and IMRaD article structures. The Evaluation of an IPSERC article might be structured according to IMRaD.

very well. These projects are also very fruitful for PhD students, who get training in supervision before they take on supervising MSc projects. By the way, the inscrutable decision by the faculty to stop new courses in 2013 was lifted a few years later.

Acknowledgments

Thank you, Pierre Nugues, for sharing your ideas on this topic with me, and for all other interesting discussions we have had. Thanks to Boris Magnusson for feedback on an earlier draft of this paper.

References

Jakob Grundström and Pierre Nugues. 2014. Using syntactic features in answer reranking. In *AAAI 2014 Workshop on Cognitive Computing for Augmented Human Intelligence*, pages 13–19. AAAI.

Magnus Norrby and Pierre Nugues. 2015. Extraction of lethal events from wikipedia and a semantic repository. In *workshop on Semantic resources and semantic annotation for Natural Language Processing and the Digital Humanities at NODALIDA 2015*, pages 28–35. Linköping University Electronic Press.

Rebecka Weegar, Linus Hammarlund, Agnes Tegen, Magnus Oskarsson, Kalle Åström, and Pierre Nugues. 2014. Visual entity linking: A preliminary study. In *AAAI-14 Workshop on Computing for Augmented Human Intelligence*.

Exhuming a Swedish Temporal Relation Dataset from the Past

Richard Johansson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
richard.johansson@gu.se

Abstract

I resurrected a dataset containing annotated event-to-event temporal relations in Swedish traffic accident reports, and then benchmarked how well contemporary large language models classify these relations. Can the current generation of LLMs outperform our results from two decades ago?

1 Introduction

My initiation as researcher in NLP took place in 2003 at Lund University in the context of Pierre's Vinnova-funded project *Development of a Text-to-Scene Converter for Vehicle Accident Reports*, where we developed an automatic system that converted traffic accident reports written in Swedish into animated sequences. In retrospect, the project had goals that seem absurdly ambitious for its time; only very recently, I encountered a paper describing an implementation of essentially the same idea, building on 2020s technology (Elmaaroufi et al., 2024). In the end, we were only able to scratch the surface of this immense task, given the state of technology at the time. However, the great variety of fundamental technical, linguistic, and philosophical challenges that we encountered in that project has influenced my research interests profoundly in the ensuing years.

The system we developed – *Carsim* (Johansson et al., 2005) – consisted of three subsystems: an NLP system, which converted the raw text into a formal representation describing the objects and events mentioned in the narrative; a scenario planner that applied physical reasoning to determine a plausible realization in the physical world of the formal representation; and finally a 3D visualizer presenting the output of the planner as an animated video. The NLP part of the Carsim system consisted of a chain of modules that carried out linguistic analysis at different levels. Except for the part-of-speech tagger, all of them were in-house implementations.

While some of these linguistic modules were rule-based domain-specific heuristics, others were based on supervised machine learning solutions where the training data was annotated by members in the group, including thesis students supervised by Pierre. For instance, Carsim's noun phrase coreference solver came out of a thesis project by Danielsson (2005), where he annotated a substantial number of coreference chains and then implemented the decision tree-based approach by Soon et al. (2001). I also annotated some of these training datasets; most importantly for the research in the later part of my PhD period, Carsim had a semantic role labeler based on a domain-specific adaptation of a small set of FrameNet frames. Pierre lent me his printed copy of a recent *Computational Linguistics* issue and pointed me to the article by Gildea and Jurafsky (2002), and we reimplemented their statistical semantic role classifier and trained it on a dataset I annotated.

For this Festschrift, I thought that it would be interesting to revisit one or more of these tasks and datasets, and see how our solutions from that time compare to what can be achieved with modern NLP techniques. For reasons I will describe below, I decided to carry out a set of experiments with a dataset annotated by Anders Berglund for his Master's thesis project (Berglund, 2004), where he considered the problem of determining *temporal relations* between events mentioned in a text. This is not only an exercise in nostalgia but also an interesting benchmarking experiment for modern large language models (LLMs), in which we can investigate their capability of reasoning about complex narratives written in Swedish.

2 Recreating the Temporal Relation Dataset

I ran into the practical difficulty that the Carsim implementation has not been published as a repository and appeared to be lost in the mists of time. However, I found a solution. When I was about to leave Lund in early 2009, Lars Nilsson was generous

From Electrophoresis to Wikidata: Festschrift in honor of Pierre Nugues, pages 11–15.
Published by University of Tartu Library, May 2025.

enough to let me keep the hard drive from my desktop computer. For some reason, I did not discard this disk and it eventually ended up among some old junk in a box in my apartment, where I recently rediscovered it. I had a SCSI-to-USB adapter and the disk had not broken down in the years since it was last used. Eventually, I was able to locate several interesting files from the past including the full directory of the Carsim implementation, with all source code and data still present.

After exploring our old files, I concluded that Berglund's temporal relation dataset was the easiest one to work with as well as the most interesting on an intellectual level. From a practical point of view, it was convenient that Berglund had used a comparatively modern stand-off annotation format, which made it possible for me to extract his data (after some mild annoyances and manual correction of token offsets). The noun phrase coreference dataset by Danielsson (2005) was also neatly formatted, but it seemed to me that analyzing temporal relations between events would be a more challenging and interesting task for a modern NLP evaluation.

3 Modeling and Annotating Temporal Relations between Events

To exemplify the annotation of temporal relations between events, consider the example below. In the example, the five events (e_1–e_5) mentioned in the text have been underlined.

> Two people died$_{e_1}$ late yesterday evening when a car drove off$_{e_2}$ the road and crashed$_{e_3}$ into a tree. The car was overtaking$_{e_4}$ another car when the driver lost control$_{e_5}$ of it.

It is worth stressing here that the events are not presented in a chronological order, and the writer first expresses the most salient information: that there were fatalities. To understand how the events are positioned in time in relation to each other, the reader has to form a mental model of what happened in the described scenario. To understand the temporal structure, the reader can also consider time expressions (*late yesterday evening*), temporal connectives (*when*) and the interplay of tenses and aspects of the verbs (*crashed*, *was overtaking*).

There are multiple conceptual frameworks for modeling events and their relations. One famous framework is by Allen (1984), who defined 13 relation types. If we apply Allen's framework to annotate the event-to-event relations in the example above, we get the structure presented in Figure 1.

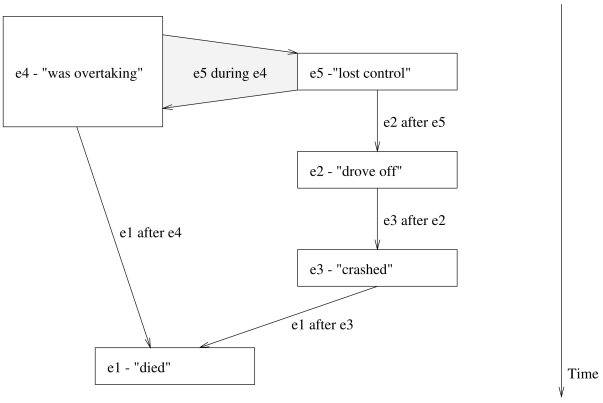


Figure 1: Allen-style relations between the events in the example text. Figure by Berglund (2004).

TimeML (Pustejovsky et al., 2003a) is one of the most widely known annotation models for annotating events and their temporal relations, and this model has been used to annotate the TimeBank corpus (Pustejovsky et al., 2003b) in English, as well as several related projects for other languages. Berglund's thesis project used a simplified TimeML annotation scheme (Berglund et al., 2006a): our project only required us to annotate relations between events that took place within the narrative described in the text. The full TimeML model can also represent e.g. hypothetical scenarios.

4 Experimental Setup

4.1 Data

After reverse-engineering Berglund's format (§2), I ended up with a dataset consisting of 27 news reports taken from a larger collection gathered in the Carsim project from various Swedish news sources, primarily *Sydsvenskan*. This corpus includes 904 annotated temporal relation instances, after discarding a few instances where the annotator was uncertain. In the experiments, I only included the 904 relations directly annotated in the dataset, and I did not expand the set of instances by applying temporal reasoning (e.g. transitivity and symmetry).

The annotation uses five temporal relation types: *after*, *before*, *includes*, *is_included*, and *simultaneous*. The most common annotated relation type (385 instances) is *before*. This reflects the fact that relations are annotated from the first event to the second, and that events are often presented somewhat chronologically in the narrative.

4.2 Selected Models

In the experiments, I evaluated a set of LLMs from the following families. They were accessed through their respective APIs without any fine-tuning.

- OpenAI's GPT models: *GPT-4o*, *GPT-4.1*,[1] and the *o3-mini* reasoning model.
- Meta's Llama models: *Llama 3 70B* (Llama Team, 2024), *Llama 3.1 405B*, and *Llama 4 Maverick*, via the Replicate API.
- DeepSeek's *V3* (chat) and *R1* (reasoning) models (DeepSeek-AI, 2025).

Before considering whether how well these models are capable of reasoning about complex scenarios, it is worth asking whether these models have any Swedish-language capabilities at all. In previous benchmarking experiments for Swedish, they have been found to perform quite well, including in a sense disambiguation exercise I carried out previously (Johansson, 2024).

4.3 Prompt Design and Model Execution

The prompts consisted of the following parts:

1. a preamble describing the temporal relation classification task and defining the five relation labels;
2. a demonstration of the classification task;
3. the full text, where the two events under consideration are surrounded by XML tags `<event1>` and `<event2>`, respectively.

The models were applied in two different settings:

- *direct* prediction: the model is given the prompt and has to output the predicted relation label directly;
- *chain-of-thought* prediction (Wei et al., 2022): the model is given the prompt and is asked to provide a textual explanation before predicting the label.

The two reasoning models (OpenAI's `o3-mini` and DeepSeek's V1) use an internal chain-of-thought process so the direct prediction approach is not applicable for those models.

Since I am primarily interested in how well the LLMs model the temporal structure of the narrative, temporal relations were predicted for individual event pairs and I did not enforce global consistency of the temporal graph (e.g. by breaking cycles).

[1]This model was released as on the day I was finishing this paper, so I had to carry out some last-minute experiments.

Model	D	CoT
`o3-mini`		0.691
`deepseek-reasoner`		0.684
`gpt-4.1`	0.486	0.640
`deepseek-chat`	0.311	0.633
`llama-4-maverick`	0.243	0.620
`llama-3.1-410b-instruct`	0.480	0.553
`gpt-4o`	0.335	0.508
`llama-3-70b-instruct`	0.327	0.362
Baseline	0.426	

Table 1: Accuracies for all models with direct prediction (D) or chain-of-thought prediction (CoT).

5 Results

5.1 Can LLMs Classify Temporal Relations?

Table 1 shows the classification accuracies for all models. This comparison also includes a trivial baseline that assumes that events are presented linearly in a chronological order: that is, it consistently predicts that the first event happens *before* the second event. This is also the majority-class baseline.

A few observations can be made about these results. First, it is clear that it is difficult for all these LLMs to classify the temporal relations *directly*: while all evaluated models outperform a uniform random baseline (0.20), only a couple of them reach the accuracy of the trivial baseline assuming a chronological order. This shows that temporal relation classification is a comparatively difficult task for LLMs; one has to be careful in drawing "cognitive" conclusions from experiments like this, but these results do not suggest that the current generation of LLMs form an internal representation of the events described in the narrative, or at least not one that easy for the model to access.

On the other hand, there are consistent improvements in the quality of predictions when the models are prompted to present their reasoning. This improvement is most notable for a couple of the more recent models (Llama 4 and DeepSeek chat), which saw low accuracies in the direct prediction setting but much higher in the chain-of-thought setting. The strongest models are the two reasoning models, which include mechanisms to improve chain-of-thought reasoning at training and inference time.

Are these results comparable to the capability of

Model	U	D
o3-mini	0.704	0.900
deepseek-reasoner	0.682	0.811
gpt-4.1	0.440	0.367
llama-3.1-410b-instruct	0.833	0.401
gpt-4o	0.610	0.181
llama-3-70b-instruct	0.714	0.103

Table 2: Consistency of predicted undirected (U) and directed (D) temporal relations.

human annotators? It is difficult to compare these predictive accuracies to human-to-human agreement levels, since Berglund (2004) did not carry out an inter-annotator agreement study. The accuracy of the best model compared to the human annotation corresponds to a Cohen's κ of 0.56, which is lower than the κ of 0.71 for relation type annotation reported for TimeBank,[2] but this comparison must of course be taken with a grain of salt.

5.2 How Consistent are the Predicted Relations?

If LLMs form some sort of representation of the events in the narrative, one would expect the predicted relations to be structurally consistent. For instance, if the model predicts that the police arrived *after* a traffic accident, it should also predict that the accident happened *before* police arrived.

To investigate the consistency of predicted relations, I switched the order of the <event1> and <event2> tags in the prompt and compared the predictions. The evaluation considers one *undirected* relation type (*simultaneous*), where labels should not change, and four *directed* types where labels should change to their inverses (e.g. *after*→*before*).

Table 2 shows the result of the consistency evaluation. The results again show that non-reasoning LLMs have rather poor temporal processing capabilities, but reasoning models are much better in this respect. In particular, the predicted directed relations are much more consistent for these models.

5.3 Can the Best Models Outperform a Decision Tree?

Berglund's thesis project also included the development of a decision tree-based classifier that predicts the type of temporal relation holding between two given events. (This work was later published as a separate EACL paper (Berglund et al., 2006b).) This classifier used a large feature set based on linguistic features of the two event mentions as well as various structural features (e.g. distances). Do current LLMs perform better than our twenty-year-old decision tree classifier?

Model	Accuracy
Carsim	0.728
o3-mini	0.728
deepseek-reasoner	0.717

Table 3: Results on the Carsim-annotated subset.

In the short time I had to write this paper, I was unable to run the Carsim implementation or to disentangle the decision tree classifiers from the rest of the code. However, I found a file where Carsim had computed temporal relation labels for event pairs in 10 of the texts in the corpus, which I could compare to Berglund's manual annotations. I then looked at the LLM predictions for the same subset.

Table 3 shows the result of this evaluation. The best LLM (o3-mini) correctly classified 134 event pairs (an accuracy of 0.728), which is exactly the same as the number of pairs correctly labeled by Carsim's classifier. I suppose this means that we can tentatively conclude that 2025 was the year when the field of NLP caught up with the work we did in Pierre's group at LTH two decades earlier.

6 Final Words

As I mentioned in the introduction, my interests as a researcher have been shaped by the research problems we encountered in this first project I worked on as a PhD student under Pierre's supervision. My current research is a bit different, but I have often thought it would be interesting to return to some of the research questions we worked on at that time, as in the little investigation carried out for this paper. I am grateful to Pierre for approaching me out of the blue after I took his course in Natural Language Processing and Computational Linguistics to suggest that I apply for his PhD position, and then for being a patient PhD supervisor in the five years after that. Those were fun years when I could be free and irresponsible and work on fascinating research problems, and along the way have interesting conversations with a free-thinking and erudite supervisor on all sorts of topics!

[2] https://timeml.github.io/site/timebank/documentation-1.2.html#iaa

References

James F. Allen. 1984. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154.

Anders Berglund. 2004. Extracting temporal information and ordering events for Swedish. Master's thesis, Lund University.

Anders Berglund, Richard Johansson, and Pierre Nugues. 2006a. Extraction of temporal information from texts in Swedish. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Anders Berglund, Richard Johansson, and Pierre Nugues. 2006b. A machine learning approach to extract temporal information from texts in Swedish and generate animated 3D scenes. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 385–392, Trento, Italy. Association for Computational Linguistics.

Magnus Danielsson. 2005. Maskininlärningsbaserad koreferensbestämning för nominalfraser applicerat på svenska texter. Master's thesis, Lund University.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Karim Elmaaroufi, Devan Shanker, Ana Cismaru, Marcell Vazquez-Chanlatte, Alberto Sangiovanni-Vincentelli, Matei Zaharia, and Sanjit A. Seshia. 2024. ScenicNL: Generating probabilistic scenario programs from natural language. In *Proceedings of the Conference on Language Modeling (COLM)*, Philadelphia, United States.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Richard Johansson. 2024. How well do large language models disambiguate Swedish words? In *Proceedings of the Swedish Language Technology Conference (SLTC)*, Linköping, Sweden.

Richard Johansson, Anders Berglund, Magnus Danielsson, and Pierre Nugues. 2005. Automatic text-to-scene conversion in the traffic accident domain. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1073–1078, Edinburgh, United Kingdom.

AI @ Meta Llama Team. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

James Pustejovsky, José Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir Radev. 2003a. TimeML: Robust specification of event and temporal expressions in text. In *AAAI Spring Symposium on New Directions in Question-Answering (Working Papers)*, pages 28–34.

James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, Daniel Day, Lisa Ferro, and Marcia Lazo. 2003b. The TIMEBANK corpus. In *Corpus Linguistics*, pages 647–656.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

On synthetic and real images as training data for object detection - A brief review

Martin Georg Ljungqvist
Axis Communications AB / Lund, Sweden
martin.ljungqvist@axis.com

Abstract

To train neural networks, sufficiently large and diverse datasets are needed. To address this, the use of synthetic data has become popular because it is inherently scalable and can be automatically annotated. A brief overview of recent work on using synthetic and real images as training data for object detection is presented in this paper, with a focus on mixing real and synthetic training data. The trend is that having real data and adding some amount of synthetic data helps the performance in many studies. It was concluded that there appears to be no consensus on the ratio of real and synthetic image data.

1 Introduction

Presented here is a brief overview of selected articles on using both real and synthetic data for object detection, the papers reviewed here cited the article by Ljungqvist et al. (2023). The overview focused on mixing real and synthetic training data.

Synthetic data here refer to images produced entirely in a computer, while real data here refer to images capturing a natural scene by a camera. Data augmentation can be applied to both synthetic and real data.

Since synthetic images and real images come from different distributions due to sensor noise, texture and many other factors, there is a domain gap and the results of training on only synthetic data and testing on only real data can produce much lower results than if training on real data (Ljungqvist et al., 2023). However, real and annotated data might not always be available in large volumes and therefore mixing real and synthetic data can gain performance.

These are the papers covered:

- Searching for the Ideal Recipe for Preparing Synthetic Data in the Multi-Object Detection Problem (Staniszewski et al., 2025)
- In the Search for the Balance Between Real and Synthetic Images in Multi-Class Detection Systems (Cecchetti et al., 2024)
- Ai-generated images as data source: The dawn of synthetic era (Yang et al., 2023)
- Bridging the gap: Active learning for efficient domain adaptation in object detection (Menke et al., 2024)

2 Mixing real and synthetic data

2.1 Searching for the Ideal Recipe for Preparing Synthetic Data in the Multi-Object Detection Problem

In Staniszewski et al. (2025) the authors generated images of 3D synthetic objects in 6 categories aimed at public transport scenarios: bicycles, trolleys, wheelchairs, boxes, suitcases, and bags.

The synthetic objects were positioned on a background collage of randomized real images from news, sports, etc.

For real data a mix of COCO and other images was used; 3 of the classes were in the COCO dataset - the data were henceforth complemented with images from a dataset for public transportation and the Internet.

The synthetic images were only added to the training set. Data augmentations were used.

They used a YOLOv7 object detector pretrained on COCO and trained it with the backbone frozen. Freezing the backbone while training on synthetic data is motivated by the study of Hinterstoisser et al. (2018). However, Tremblay et al. (2018) showed promising results for unfrozen backbone using domain randomized synthetic data. Ljungqvist et al. (2023) reported that

From Electrophoresis to Wikidata: Festschrift in honor of Pierre Nugues, pages 16–19.
Published by University of Tartu Library, May 2025.

no particular difference could be seen in performance for frozen or unfrozen backbone. It seems that there is not a consensus whether freezing the backbone or not is preferred for training on synthetic data.

It should be noted that this study claimed that synthetic training data "significantly improves results", however there was no mention of multiple trainings, t-test, or p-values which are needed to test statistically significant differences. However, they used cross-validation for model selection.

They trained the detector on a mix of real and synthetic data using different ratios between real and synthetic training data.

Using only synthetic and no real data gives very low performance. Adding only 5% real data gives a tremendous boost.

The results showed a trend that adding synthetic data clearly improved the results. They observed a boost in performance when adding 25% or more synthetic data in relation to the amount of real data. Soon after 25% added synthetic data the performance flattened out.

Having real data and adding any amount of synthetic data increased the performance of all metrics in this study. Adding synthetic data did not decrease the performance except for some cases of the objectness metric. The objectness and classification metrics had more variance compared to the other metrics.

The best results without transfer learning were when having slightly more synthetic than real data, the best mean average precision at 0.50 intersection of union (mAP50) was for 1 real and 1.75 synthetic data, but it varied for different metrics between 1:1 and 1:2.

With transfer learning, the best mAP50 was for the ratio of 1 real and 0.75 synthetic data, but it varied for different metrics between 1:0.25 and 1:1.25.

In this study, the influence of the amount of synthetic data relative to real data was concluded to be minimal. However, it is worth noting that synthetic data can be a good replacement and complement to real data when real data is scarce and there is a need to balance data from multiple classes.

This study reported interesting results on the topic of mixing real and synthetic image data. However, the study had limitations such that only one synthetic dataset was used, and there was no statistical evaluation of differences in the results, such as error bars or t-test. Furthermore, balancing multiple classes may not always be strictly needed.

2.2 In the Search for the Balance Between Real and Synthetic Images in Multi-Class Detection Systems

A literature review on mixing real and synthetic images in 27 articles from the past 5 years was performed in Cecchetti et al. (2024).

Only three of the articles in the review presented a specific ratio; however, they did not present a comparison between the different ratios and their impact on object detection performance.

This literature review observed that most of the articles did not establish a relationship between the use of real and synthetic images. They conclude that there is a lack of in-depth analysis of the ratio relationship of mixing real and synthetic data for multi-class detection tasks. Hence, there is currently a scientific gap in terms of the number of synthetic images used to train object detection models.

It should be noted that the Staniszewski et al. (2025) study was published after this literature review was performed.

2.3 AI-generated images as data source: The dawn of synthetic era

In the review paper Yang et al. (2023) results were included for object detection results from Ge et al. (2022), and extended in Ge et al. (2023), for Pascal VOC and COCO detection reporting mAP50 and mAP. They used Faster RCNN with ResNet-50 backbone. Combining real and synthetic images (copy paste and foreground) resulted in higher mAP50 and mAP for both Pascal VOC and COCO.

They combine real and synthetic data in various ways using either synthetic foreground or background, so there is no specified ratio of distinct real and synthetic images.

On Pascal VOC, a model trained solely on synthetic data in the absence of any real images achieves comparable performance to a model trained on 1,464 real images. Furthermore, adding synthetic backgrounds when blending both real and synthetic led to the best performance.

For the COCO dataset, mixing both synthetic and real gave the greatest performance boost.

Hence, the conclusion is that mixing real and synthetic data gives a boost.

2.4 Bridging the gap: Active learning for efficient domain adaptation in object detection

Menke et al. (2024) combined real and synthetic data for training using active learning for domain adaptation. They used the synthetic sim10k dataset as the source domain and the natural images in the Cityscapes dataset as the target domain, in a synthetic-to-real domain adaptation setting. The aim was to adapt the object detection models trained on sim10k to perform well on Cityscapes. The objective was to select a portion of the target domain for labeling (aiming to outperform unsupervised domain adaptation methods that rely solely on unlabeled target data).

Sim10k contains 10,000 images from the computer game Grand Theft Auto 5, the paper did not specify a split for these data, so it is assumed that they used all for training. Cityscapes contains 2975 labeled images in the training set, and they used up to 25% of this for training, which results in approximately 743 images, which is about 7% of 10,000.

The mix of synthetic and real training data is performed using an active learning approach; real images from Cityscapes are selected for training with the aim of complementing the source domain (sim10k). This was performed by scoring the images using a scoring function to decide if the image should be used for training. In this study, three different scoring functions were proposed and evaluated. They used 25% of the target domain training images.

They show mAP performance on the percentage of labeled images (from 5% to 25%) and the number of sampled boxes (from about 1,000 to 12,000). The absolute mAP increased with increasing number of labeled target images and with the number of labeled boxes. This is interpreted in such a way that they used a fixed amount of synthetic training data, and for an increasing amount of real images the performance increased. The amount of real data was in all ratios smaller than the synthetic amount. Using 25% of the target data, the real data consists of about 7% of the amount of synthetic data.

Using 25% labeled target domain images, there was a 2.43 mAP improvement in object detection performance over the random baseline.

3 Discussion and Conclusion

From this brief literature review, it was concluded that there seems to be no consensus on the ratio of real and synthetic image data.

None of the studies had a statistical evaluation of differences in the results, such as standard deviation, error bars, or t-test.

The trend is that having real data and adding some amount of synthetic data helps the performance in many studies.

There is probably no ratio that works for everything; it could depend on factors such as:

- Data quality
- The class types
- Variations within the classes
- Domain gap factors
- Amount of real data available
- How the synthetic data is generated

And other factors may likely come into account here. There is no one-size-fits-all for mixing synthetic and real images as training data.

Acknowledgments

The author wishes to thank colleagues for valuable discussions on the topic.

References

Vitória Biz Cecchetti, Marcelo Rudek, and Roberto Z. Freire. 2024. https://doi.org/10.1109/ICIEA61579.2024.10664977 In the search for the balance between real and synthetic images in multi-class detection systems. In *2024 IEEE 19th Conference on Industrial Electronics and Applications (ICIEA)*, pages 1–6.

Yunhao Ge, Jiashu Xu, Brian Nlong Zhao, Neel Joshi, Laurent Itti, and Vibhav Vineet. 2022. http://arxiv.org/abs/2206.09592 Dall-e for detection: Language-driven compositional image synthesis for object detection.

Yunhao Ge, Jiashu Xu, Brian Nlong Zhao, Neel Joshi, Laurent Itti, and Vibhav Vineet. 2023. http://arxiv.org/abs/2309.05956 Beyond generation: Harnessing text to image models for object detection and segmentation.

Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. 2018. http://arxiv.org/abs/1710.10710 On pre-trained image features and synthetic images for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*.

Martin Georg Ljungqvist, Otto Nordander, Markus Skans, Arvid Mildner, Tony Liu, and Pierre Nugues. 2023. Object detector differences when using synthetic and real training data. *SN computer science*, 4(3):302.

Maximilian Menke, Thomas Wenzel, and Andreas Schwung. 2024. https://doi.org/https://doi.org/10.1016/j.eswa.2024.124403 Bridging the gap: Active learning for efficient domain adaptation in object detection. *Expert Systems with Applications*, 254:124403.

Michał Staniszewski, Aleksander Kempski, Michał Marczyk, Marek Socha, Paweł Foszner, Mateusz Cebula, Agnieszka Labus, Michał Cogiel, and Dominik Golba. 2025. https://doi.org/10.3390/app15010354 Searching for the ideal recipe for preparing synthetic data in the multi-object detection problem. *Applied Sciences*, 15(1).

Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. 2018. http://arxiv.org/abs/1804.06516 Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Autonomous Driving*.

Zuhao Yang, Fangneng Zhan, Kunhao Liu, Muyu Xu, and Shijian Lu. 2023. http://arxiv.org/abs/2310.01830 Ai-generated images as data source: The dawn of synthetic era.

Universal Dependencies are neither Universal nor Dependencies

Joakim Nivre
Uppsala University
Department of Linguistics and Philology
joakim.nivre@lingfil.uu.se

Abstract

Universal Dependencies is a framework for cross-linguistically consistent treebank annotation, using binary relations to represent syntactic structure. Despite the name of the framework, it should not be assumed that the relation types are all universal, nor that they are all asymmetric dependency relations in a narrow sense. The purpose of this paper is to explain why and to clear up a few common misconceptions about Universal Dependencies.

1 Introduction

Universal Dependencies (UD) is a framework for morphosyntactic annotation, designed to be applicable to all human languages and to enable meaningful cross-linguistic comparisons. The two versions of the guidelines are described in Nivre et al. (2016) and Nivre et al. (2020); a longer description of the underlying linguistic theory can be found in De Marneffe et al. (2021); and annotated data for 168 languages[1] can be found together with additional documentation on the UD website.[2]

One of the basic design principles of UD is that syntactic structure can be represented by trees that are composed of binary syntactic relations. By way of example, Figure 1 shows a UD representation for the English sentence *Sherlock Holmes will solve the mystery in spite of the strange clues.* The syntactic representation is a tree rooted at the word *solve*, and all edges of the tree are labeled with syntactic relation types such as nominal subject (*nsubj*), direct object (*obj*), and so on. In addition, each word is assigned a part-of-speech tag like NOUN, VERB, ADJ, etc.[3]

The syntactic relations used in UD often correspond to the asymmetric head-dependent relations that are central to the dependency grammar tradition, but UD does not assume that all syntactic relations are dependency relations in this sense. This has led to some criticism of the UD framework from proponents of dependency grammar, based on the observation that UD relations sometimes appear to violate commonly held assumptions among dependency grammarians (see, for example, Gerdes and Kahane, 2016; Gerdes et al., 2018; Osborne and Gerdes, 2019). The first point of this paper is to clarify the status of syntactic relations in UD in this respect.

UD is designed to be applicable to all natural languages, and the same should hold of the inventory of syntactic relation types posited in UD. However, this does not mean that all relations are assumed to be universal in the sense that they can be found in all languages. Clarifying this issue and relating it to research in linguistic typology is the second point of this paper. Putting the two points together gives us the apparent contradiction in the title of the paper, which we will now try to resolve.

Before proceeding, however, I want to clarify that the views expressed in this paper are my own and sometimes go beyond the official UD documentation. I have arrived at these views by reflecting on the evolution of UD, as well as on the criticism that it has received, and this has helped me deepen my understanding of UD. My hope is that readers may find it somewhat helpful as well.

2 UD Relations are not Dependencies

Dependency grammar is a family of syntactic theories and frameworks, which all assume that an important part of natural language syntax can be analyzed in terms of binary asymmetrical relations between a *head* and a *dependent*. Such re-

[1] UD v2.15, released November 15, 2024.

[2] https://universaldependencies.org

[3] In the full UD representation, each word is also assigned a lemma and a set of morphological features, which have been suppressed in Figure 1 for readability.

From Electrophoresis to Wikidata: Festschrift in honor of Pierre Nugues, pages 20–24.
Published by University of Tartu Library, May 2025.

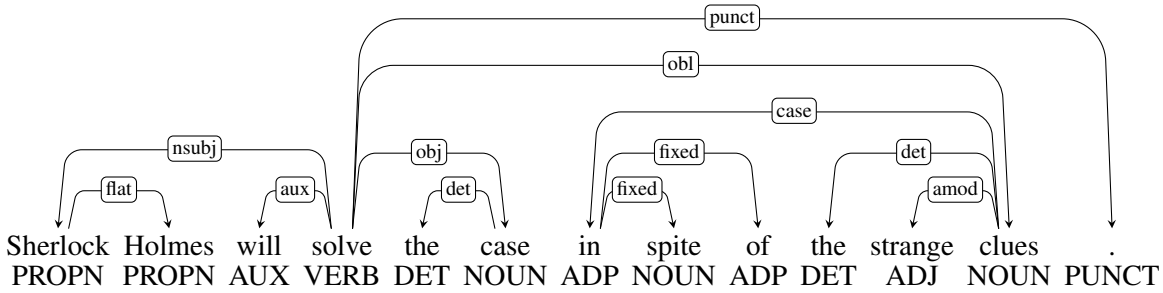


Figure 1: Simplified UD representation of an English sentence.

lations are called *dependency* relations, because of the asymmetric relation, where the dependent in some way presupposes the head, but not vice versa. However, it has been hard to reach consensus on the criteria for distinguishing dependency relations, and for identifying the head and dependent in different constructions (see, for example, Nivre, 2006, chapter 3).

The basic idea underlying dependency grammar has been traced back to the Indian grammarian Pāṇini several centuries before the common era, and modern elaborations of this idea can be found in Tesnière (1959), Hudson (1984), Sgall et al. (1986), and Mel'čuk (1988), among others. Theories in this family all agree on the central role of dependency relations, but differ in many of the details, and not all of them assume that syntactic structure can be analyzed only in terms of dependency relations. UD follows Tesnière (1959) and Hudson (1984) in *not* assuming that all syntactic relations are dependency relations.

Of the 37 syntactic relations in the UD inventory,[4] the ones that clearly fulfill the criteria of dependency relations are argument and modifier relations, which are also the relations which all varieties of dependency grammar tend to agree on. Thus, in Figure 1, the subject (*nsubj*) and object (*obj*) relations are dependency relations that hold between the head verb *solve* and the dependent arguments *Sherlock Holmes* and *the case*, respectively. Similarly, the oblique modifier (*obl*) relation relates the head verb *solve* to the dependent *in spite of the confusing clues*, and the adjectival modifier (*amod*) relates the head noun *clues* to the dependent *strange*. But the remaining relations are not straightforward dependency relations.

Perhaps the most obvious example of a non-dependency relation in UD is the relation called *flat*, which, as the name suggests, is used to connect two elements of a sentence that belong together but where neither of them can be identified as the head. Hence, it denotes a *symmetric* relation between two elements, not an asymmetric dependency relation. An instance of the *flat* relation can be found in Figure 1, where the name *Sherlock Holmes* is analyzed as a headless phrase, because standard tests for dependency does not clearly identify either the first or the last name as the syntactic head of the phrase.[5] Another clear non-dependency relation is the *fixed* relation, which combines words that were historically connected by syntactic relations but which have frozen into a fixed expression, such as *in spite of* in Figure 1, which functions as a preposition introducing an oblique modifier despite on the surface being a preposition-noun-preposition sequence.

Slightly more controversial are the relations holding between content words and function words, such as the auxiliary (*aux*) relation from *solve* to *will*, the determiner (*det*) relation from *case* to *the* (and from *solutions* to *the*), and the case marker (*case*) relation from *solutions* to *in spite of*. UD follows Tesnière (1959) in assuming that such relations are not really dependency relations,[6] but instead are relations that allow multiple words to form syntactic nuclei in a way analogous to morphological inflection of content words. Thus, the function words involved in *aux*, *det* and *case* relations often correspond to morphological affixes in other languages, or even in the same language.

The UD treatment of function words has been the subject of some debate (see, for example, Gerdes and Kahane, 2016; Gerdes et al., 2018;

[4] https://universaldependencies.org/u/dep/index.html

[5] One such test is substitutability of the head for the whole phrase, and in this case both *Sherlock* and *Holmes* can replace the whole phrase without loss of grammaticality or drastic change of meaning.

[6] For example, they fail the usual substitution test in that none of the elements can replace the whole.

Osborne and Gerdes, 2019), because it appears to violate the assumption that function words are syntactic heads, which is found in many versions of dependency grammar (as well as syntactic theories in other traditions). In my view, this criticism rests on a misconception, because UD does not assume that function words are dependents of content words, nor the other way around. Instead, content words together with their associated function words form the syntactic units that enter into dependency relations (and other syntactic relations).[7]

To be fair, it is not hard to see how these misconceptions may arise. Besides the name of the framework, the nature of the tree-shaped syntactic representations,[8] where every word except one is attached to another word with a directed edge, may give the impression that every word except one is a syntactic dependent of another word. In addition, the UD guidelines have not always been clear about the status of syntactic relations, often using the terms *head* and *dependent* as convenient shorthands for *parent node* and *child node* in the syntactic tree. These tree-shaped representations are a heritage from the parsing community from which UD originates and are better described as *spanning trees*, a term used in the graph-based approach to dependency parsing (McDonald et al., 2005a,b; McDonald and Pereira, 2006). This term only implies that every word of a sentence is included in the tree, but remains neutral about the nature of the pairwise syntactic relations. These relations are instead specified by syntactic relation types, encoded in labels like *nsubj*, *obj*, *flat*, and *fixed*. And while some of these relation types fulfill the criteria of dependency relations, others clearly do not. Hence, UD relations are not (all) dependencies.

3 UD Relations are not Universal

Syntactic relation types in UD should be cross-linguistically applicable, meaning that it should be possible to identify instances of them without relying on language-specific criteria. However, this does not mean that the relations are universal in the stronger sense that they are assumed to exist in all languages – at least not all of them.

To explain this distinction, it may be helpful to introduce some concepts from linguistic typology. Croft (2022) distinguishes two types of comparative concepts, which can be used when comparing languages typologically:

> **construction:** any pairing of form and function in a language (or any language) used to express a particular combination of semantic content and information packaging

> **strategy:** a construction in a language (or any language), used to express a particular combination of semantic content and information packaging (the 'what'), that is further distinguished by certain characteristics of grammatical form that can be defined in a crosslinguistically consistent fashion (the 'how')

A construction, in Croft's sense, is universal and defined only in terms of its function (which in turn is defined in terms of semantic content and information packaging), while a strategy is a particular way of realizing this function morphosyntactically, and hence not universal.

To exemplify these concepts, let us consider the *predicate nominal* construction, which is "a clause construction defined by the function of predicating an object concept of a referent – that is, asserting what object category the referent belongs to".[9] Two common strategies for this construction are exemplified in (1) and (2–3).

(1) Иван танцор
Ivan.NOM dancer.NOM
'Ivan is a dancer'

(2) Ivan är dansare
Ivan COP dancer
'Ivan is a dancer'

(3) Ivan is a dancer
Ivan COP a dancer

The Russian example in (1) uses a *zero* strategy (Stassen, 1997), which simply juxtaposes the referring expression Иван with the noun танцор in nominative case expressing the object concept. By contrast, the Swedish and English examples in (2) and (3) both use a *verbal copula* strategy (Stassen, 1997), where predication is mediated by a copula verb. The notion of strategy allows us to abstract

[7] An elaboration of the UD position on function words can be found in De Marneffe et al. (2024). An overview of the treatment of function words in different versions of dependency grammar can be found in Osborne (ed.) (2024).

[8] The tree constraint holds for *basic* UD representations; there is also an *enhanced* representation, which takes the form of a general graph, which will not be discussed here.

[9] https://comparative-concepts.github.io/cc-database/

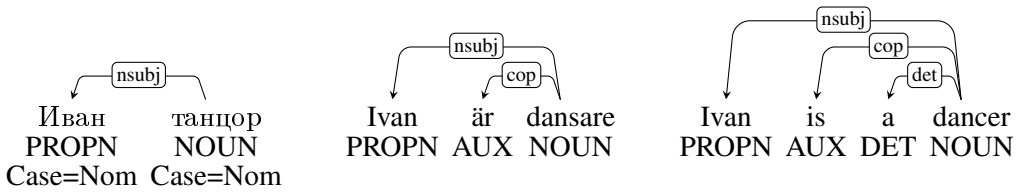


Figure 2: Simplified UD annotation for predicate nominal constructions in Russian, Swedish and English.

over language-specific constructions and say that Swedish and English use the same strategy, while the Russian strategy is different.

UD needs to annotate elements of both constructions and strategies, but since it is a framework for morphosyntactic annotation, it must often give priority to strategies. Figure 2 shows how UD can represent the predicate nominal examples in Russian, Swedish, and English. What is common to all three languages is the subject (*nsubj*) relation from the nominal predicate to the referent, which can therefore be said to capture the core of the predicate nominal construction (even though it is also found as a component of other constructions). By contrast, the copula (*cop*) relation is found only in Swedish and English, and is clearly an element of a non-universal strategy.

By and large, argument and modifier relations tend to be part of constructions and are therefore universal. For example, all languages have intransitive and transitive clauses, the arguments of which instantiate subject (*nsubj*) and object (*obj*) relations, although the way that these relations map on to morphosyntactic markers varies across languages, with nominative-accusative and ergative-absolutive alignment as the main strategies. Similarly, all languages can express property modification and object modification, involving the adjectival (*amod*) and nominal (*nmod*) relations, respectively, although the morphosyntactic encoding by means of case markers or indexation again varies depending on strategies.

At the other end of the spectrum, we find relations that denote specific strategies, or strategy elements, and which are therefore not universal. These include the copula (*cop*) relation discussed above, as well as several of the function word relations discussed in Section 2. Thus, the case marker (*case*) relation is used for case markers (including adpositions) that are realized as free morphemes, a strategy that is not universal. Similarly, not all languages encode tense, aspect, mood, or evidentiality using independent words, which is what the auxiliary (*aux*) relations is used for. Hence, UD relations are not (all) universal.

4 Conclusion

In this paper, I have tried to explain why it is a mistake to assume that all syntactic relations in UD are dependency relations in the narrow sense and that they are universal in a strong sense. It follows that *Universal Dependencies* must be treated as a proper name, not as a descriptive phrase, which is why the title of this paper is not a contradiction.

Acknowledgments

The work presented in this paper was supported by Swedish Research Council grant no. 2022-02909. I am grateful to the editor for comments on the first draft of the paper.

References

William Croft. 2022. *Morphosyntax: Constructions of the World's Languages*. Cambridge University Press.

Marie-Catherine De Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47:255–308.

Marie-Catherine De Marneffe, Joakim Nivre, and Daniel Zeman. 2024. Function words in Universal Dependencies. *Linguistic Analysis*, 43(3–4):549–588.

Kim Gerdes, Bruno Guillaume, Sylvain Kahane, and Guy Perrier. 2018. SUD or Surface-Syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 66–74.

Kim Gerdes and Sylvain Kahane. 2016. Dependency annotation choices: Assessing theoretical and practical issues of universal dependencies. In *Proceedings of LAW X – The 10th Linguistic Annotation Workshop*, pages 131–140.

Richard A. Hudson. 1984. *Word Grammar*. Blackwell.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 81–88.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530.

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Dan Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*, pages 1659–1666.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Dan Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC)*, pages 4034–4043.

Timothy Osborne and Kim Gerdes. 2019. The status of function words in dependency grammar: A critique of Universal Dependencies (UD). *Glossa*, 4(1):17.

Timothy Osborne (ed.). 2024. The status of function words in dependency grammar. *Linguistic Analysis*, 43(3–4).

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Pragmatic Aspects*. Reidel.

Leon Stassen. 1997. *Intransitive Predication*. Oxford University Press.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Editions Klincksieck.

An Appendix to Pierre Nugues's Python Book

Aarne Ranta
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
aarne.ranta@cse.gu.se

Abstract

The third edition of Pierre Nugues's book uses Python to introduce several state-of-the-art techniques in data-driven NLP. This paper describes a few examples of symbolic methods using the Python bindings of Grammatical Framework, GF.

1 Introduction

Pierre Nugues gave me a draft of his book *An Introduction to Languge Processing in Perl and Prolog* (Nugues, 2006) before its publication. I liked the book very much. It covered much of the same ground as the most popular textbook in the field, (Jurafsky and Martin, 2000). But it was shorter (althouch still long) and more precise about the mathematical foundations. It also had much more to say about other languages than English. I chose it as the recommended book in my teaching.

One thing about the book that I did not use were the code examples. Perl and Prolog were still the state of the art at the time, but I preferred Haskell, a functional programming language with static typing and powerful abstraction mechanisms. Fortunately, combining Nugues's book with programming in Haskell was not a problem, because all the important concepts were explained independently of the code examples.

The third edition of the book, from 2024, changed the title: *Python for Natural Language Processing* (Nugues, 2024). Also much of the content has changed: it is now entirely about statistics and machine learning, and it covers the latest techniques such as large language models and transformers. The mathematics behind these methods is still carefully explained in separation from the code, but the code is very useful as it uses state-of-the art Python libraries that enable realistic coding exercises and experiments. I am very much looking forward to studying this book in more depth to learn new things.

Haskell is still my preferred language for research projects, but Python is easier to deploy in areas where it has library support. Python has also become a lingua franca for students and researchers in many areas, which makes it the language of choice in projects that want to attract community contributions. Last but not least, Python has over the years adopted features of functional programming, often inspired by Haskell: lambda abstractions, higher-order functions, comprehensions, and structural pattern matching.

Even at my university, Python has become the programming language number one in teaching. In the last five years, I have taught several editions of three different courses that use Python, to thousands of students. In this connection, I have written several pieces of code where I think Python works well. One of these tasks is among those that Pierre has left out from the third edition: syntax. Hence a humble — or actually rather pretentious — proposal arises to complement Pierre's book with some grammar-based applications in Python. Complete code examples can be found in `https://github.com/GrammaticalFramework/comp-syntax-gu-mlt/tree/main/python`.

2 Background: GF

Perhaps the most widely used tool for syntax in Python is NLTK, Natural Language ToolKit (Bird et al., 2009). Nugues (2024) mentions NLTK three times, about other things than syntax. NLTK's CFG class supports context-free grammars, whereas the CCG class supports combinatory categorial grammars.

Grammatical Framword (GF, Ranta, 2011) is a grammar formalism based on a Logical Frame-

From Electrophoresis to Wikidata: Festschrift in honor of Pierre Nugues, pages 25–29.
Published by University of Tartu Library, May 2025.

work (LF), which is a generic name for computer systems based on constructive type theory (Martin-Löf, 1984). GF uses LF to define **abstract syntax**, which is a free algebra of **abstract syntax trees**. It extends LF with a layer of **concrete syntax**, which is a mapping from abstract syntax trees to strings in some actual language. These mappings are by design **reversible**, which makes GF grammars usable for both **linearization** (from trees to strings) and **parsing** (from strings to trees).

An abstract syntax in GF can be equipped with several concrete syntaxes, which results in a **multilingual grammar**. Combining parsing with one concrete syntax and linearization with another one results in **translation** betwee the languages. To make this possible, GF has an expressive power slightly above CCG but still in the mildly context-sensitive class equivalent to PMCFG (Parallel Multiple Context-Free Grammar, Seki et al., 1991; Ljunglöf, 2004).

To give an example, consider the following rules of context-free grammar. They are written in the BNF (Backus-Naur Form) notation, which is also recognized by GF as a special case of full GF grammars. Each rule can be given a name; otherwise, GF creates names automatically, but they are clumsier to use.

```
Pred.  S  ::= NP VP
Compl. VP ::= TV NP
```

This grammar is in GF separated into an abstract and a concrete syntax that look as follows:

```
-- abstract
fun Pred : NP -> VP -> S
fun Compl : TV -> NP -> VP

-- concrete
lin Pred np vp = np ++ vp
lin Compl tv np = tv ++ np
```

This grammar corresponds to the SVO word order (Subject-Verb-Object). However, just changing one rule in the concrete syntax defines the SOV order:

```
lin Compl tv np = np ++ tv
```

The VSO order requires a more radical change: splitting the VP into a **record** with a verb part and an object part:

```
lin Pred np vp =
  vp.verb ++ np ++ vp.obj
lin Compl tv np =
  {verb = tv ; obj = np}
```

This is an example where the "multiple" part of PMCFG is used: instead of a single string, a VP consists of two strings, which can be taken apart as a **discontinuous constituent**. Another example is **inflection tables**. For instance, French verbs are modelled by tables that contain 53 strings:

```
table {
  VF Ind Pres Sg P1 => "aime" ;
  VF Ind Pres Sg P2 => "aimes" ;
  ...
  VF PastPart Pl Fem => "aimées"
  }
```

This table can be used as the linearization of the same abstract object as English "love", which is a table with just five forms.

Even with the expressive power of a formalism such as GF, writing grammars requires both time and expertise. The following assets help solve these problems: a **Resource Grammar Library** (RGL), which makes the grammar rules of over 40 languages usable via a high-level API (Application Programming Interface), and **embedded grammars**, program libraries that enable the access to GF grammars in main-stream programming languages such as C, Java, and Python. Embedded grammars use a runtime format of GF, called PGF (Portable Grammar Format, Angelov et al., 2009), which can be manipulated with a runtime system written in C (Angelov, 2011). The C runtime can be imported in other programming languages via their foreign function interfaces — a technique that is extensively used in Python.

Another direction in GF has been compound systems with data-driven techniques such as neural dependency parsing. Some of these projects are outlined in (Ranta et al., 2020).

3 The PGF Library in Python

To enable PGF in Python, you just need to install the pgf library:

```
pip3 install pgf
```

To test this, one can get started with a ready-made PGF file from `https://www.grammaticalframework.org/~aarne/ResourceDemo.pgf.gz`. Uncompress it and create a file `trans.py` with the following Python script, which is a simple translator loop receiving input in English and

converting it to all other languages. We have left out all "unnecessary" parts such as error handling.

```
import pgf

gr = pgf.readPGF('ResourceDemo.pgf')
inlang = gr.languages['ResourceDemoEng']

while True:
    string = input('> ')
    parses = inlang.parse(string)
    _, tree = parses.__next__()
    print(tree)
    for _, cnc in gr.languages.items():
        print(cnc.linearize(tree))
```

The script loads a PGF file with the function `pgf.readPGF()`. After that, it sets the input language to English; the keys in the `languages` dictionary are concrete syntax module names, and the values are the concrete syntaxes themselves. An infinite loop reads user input that it tries to parse in English, with the `parse()` method of the concrete syntax. A successful parse returns an iterator, here `parses`, which consists of pairs of probabilities and abstract syntax trees. The parses are returned lazily in the order of decreasing probability (which can be defined for each GF grammar separately). Here, we just print the linearizations of the first tree in each language using its `linearize()` method. Running the script looks as follows:

```
$ python3 trans.py
> this grammar knows forty languages
...
hierdie grammatika ken veertig tale
aquesta gramàtica sap quaranta llengües
denne grammatik kender fyrre sprog
see grammatika tunnab nelikümmend keelt
gramatika honek berrogei hizkuntzak ditu
tämä kielioppi tuntee neljäkymmentä kieltä
cette grammaire connaît quarante langues
diese Grammatik kennt vierzig Sprachen
...
```

Obviously, many more things can be done with the few methods shown in the above script. The full API of the PGF library is given in `https://www.grammaticalframework.org/doc/runtime-api.html#python`, explaining functionalities such as morphological analysis and tree visualization. All of these are accessible from Python without writing any GF code and even without installing the GF compiler. The compiler is, however, needed if you want to build your own .pgf files. The compiler is run on a set of .gf files as follows:

```
gf -make MyEng.gf MyFre.gf ...
```

with the list of all those concrete syntaxes that implement the abstract syntax `My`. The resulting file `My.pgf` can be used in Python programs in the way shown in the script above.

Translation is the most straightforward application of GF grammars. But it is no longer as popular as it used to be, after all advances in neural machine translation. Two application that can more clearly benefit from the underlying tree structures are **semantics** and **natural language generation**. Let us briefly cover those applications in the following sections.

4 Semantics

A widely used, traditional way of defining semantics is by **pattern matching on abstract syntax trees** (Van Eijck and Unger, 2010). It originates in Montague semantics (Montague, 1974), where it works on syntactic structures similar to those used in the GF Resource Grammar Library. Such general syntax-based semantics is probably not a very common NLP task these days. But the same principles can be applied for more specific tasks, such as query languages, which compute answers to questions by similar semantic rules.

Using a GF grammar for semantics makes it possible to share the semantics among different concrete languages, because the semantic functions operate on the abstract syntax. Let us illustrate this with a fragment of Montague's famous PTQ fragment (Proper Treatment of Quantification in Ordinary English). The most important rules are the ones having to do with predication and complementation, especially in the presence of quantifiers. We can cover an interesting part of Montague's PTQ with the following abstract syntax functions in GF:

```
Pred  : NP -> VP -> S
Compl : TV -> NP -> VP
Intr  : IV -> VP
Every : CN -> NP
Named : PN -> NP
```

For each of the involved categories, there is a **semantic type**, which in a functional notation gives the following types of semantic interpretation functions:

```
semS  : S -> bool
semPN : PN -> ind
semVP : VP -> ind -> bool
semIV : IV -> ind -> bool
semTV : TV -> ind -> ind -> bool
semCN : CN -> set
semNP : NP -> (ind -> bool) -> bool
```

Noun phrase (NP) rules are of special interest: because NP includes both proper names (PN) and quantified phrases (Every), its semantic type cannot be just `ind` of individuals. Thus, in the rule for `Pred`, it is the (semantics of) the NP that is applied to VP, not the other way round. This is natural for quantifier phrases but means that PN has to be "raised" into a function of expected type; this was one of the most famous tricks in Montague's semantics.

In Python, the semantic types can be expressed as type hints of semantic functions, which operate recursively on abstract syntax trees for each category, as shown in the code below. The only new function needed from the PGF library is the method `unpack()`, which returns a pair of a function and a list of arguments. By using the `match` statements of Python, we can write pattern matching very much like in Haskell (Van Eijck and Unger, 2010).

```
def semNP(tree: pgf.Expr) ->
      Callable[
        [Callable [[ind], bool]],
        bool]:
    match tree.unpack():
      case ('Named', [pn]):
        return lambda P: P (semPN(pn))
      case ('Every', [cn]):
        return lambda P: all(P(x)
                    for x in semCN(cn))

def semS(tree: pgf.Expr) -> bool:
    match tree.unpack():
        case ('Pred', [np, vp]):
            return semNP(np)(semVP(vp))
```

5 Natural Language Generation

Natural Language Generation (NLG) is probably the most popular application of GF grammars. Rule-based NLG in the style of Reiter and Dale (2000) is still an efficient and technique method to convert data into text. It does not "hallucinate", and it can be implemented simultaneously for a large set of languages via abstract syntax, guaranteeing that all generated languages express exactly the same content.

Abstract Wikipedia (Vrandečić, 2021; Ranta, 2023; Angelov et al., 2025) is an ongoing project where GF and Python are used for NLG. Abstract Wikipedia is based on Wikidata, a fact database developed to support Wikipedia (Vrandečić and Krötzsch, 2014). Queries to this database can be stored in JSON files, which are easy to manipulate in Python. For example, Wikidata about cities can contain entries of the following kind:

```
"Q2167": {
  "labels": {
      "en": "Lund",
     "fr": "Lund",  ...
      },
  "country": "Q34",
  "population": "98308",
  "university": "Q218506",
  "domain": "city"
  }
```

The identifiers starting with "Q" are unique identifiers of Wikidata objects, also usable as abstract syntax functions in GF. Every object may have "labels" in different languages, and concrete syntax linearizations can be derived from them. Any of these pieces of information can be missing, which is represented as a `null` value in the object. The following code generates one-line descriptions of cities from `city_data`.

```
def city_descr(city, prop=None):
    country = city_data[city][country]
    if country and property:
      tree = f'Descr prop (Loc {country})'
    elif country:
      tree = f'Descr city (Loc {country})'
    else:
      tree = 'city'
    return pgf.readExpr(tree)
```

Descriptions are generated as abstract syntax trees, which can be linearized to every supported language. A simple way to build such trees is to use f-strings for them, with slots for the country of location and the most important properties. This produces strings, from which trees can be constructed with the `readExpr()` function. The resulting description of Lund says that it is a university town in Sweden, because both `country` and `university` fields have non-null values:

```
Descr UniversityTown (Loc Q34)
```

This tree linearizes to strings such as

a university town in Sweden
une ville universitaire en Suède
yliopistokaupunki Ruotsissa

Such descriptions can typically be found as the first sentences of Wikipedia articles. The generation of complete articles is discussed in Angelov et al. (2025).

References

K. Angelov. 2011. *The Mechanics of the Grammatical Framework*. Ph.D. thesis, Chalmers University of Technology.

K. Angelov, B. Bringert, and A. Ranta. 2009. PGF: A Portable Run-Time Format for Type-Theoretical Grammars. *Journal of Logic, Language and Information*, 19:201–228.

Krasimir Angelov, Andrea Carrión del Fresno, Ekaterina Voloshina, and Aarne Ranta. 2025. Leveraging grammatical framework and wordnet for natural language generation from wikidata. In *Distributed Computing and Artificial Intelligence, Special Sessions I, 21st International Conference*, pages 173–184, Cham. Springer Nature Switzerland.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit*. O'Reilly.

D. Jurafsky and J. Martin. 2000. *Speech and Language Processing*. Prentice Hall.

P. Ljunglöf. 2004. *The Expressivity and Complexity of Grammatical Framework*. Ph.D. thesis, Dept. of Computing Science, Chalmers University of Technology and Gothenburg University.

P. Martin-Löf. 1984. *Intuitionistic Type Theory*. Bibliopolis, Napoli.

R. Montague. 1974. *Formal Philosophy*. Yale University Press, New Haven. Collected papers edited by Richmond Thomason.

Pierre Nugues. 2006. *An Introduction to Languge Processing in Perl and Prolog*. Springer.

Pierre Nugues. 2024. *Python for Natural Language Processing*. Springer.

Aarne Ranta. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford.

Aarne Ranta. 2023. Multilingual text generation for abstract wikipedia in grammatical framework: Prospects and challenges. In *Logic and Algorithms in Computational Linguistics 2021 (LACompLing2021)*, pages 125–149, Cham. Springer Nature Switzerland.

Aarne Ranta, Krasimir Angelov, Normunds Gruzitis, and Prasanth Kolachina. 2020. Abstract Syntax as Interlingua: Scaling Up the Grammatical Framework from Controlled Languages to Robust Pipelines. *Computational Linguistics*, 46(2):425–486.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

H. Seki, T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

Jan Van Eijck and Christina Unger. 2010. *Computational semantics with functional programming*. Cambridge University Press.

Denny Vrandečić. 2021. Building a Multilingual Wikipedia. *Communications of the ACM*, 64(4):38–41.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Author Index